НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “ Комп’ютерні науки ”

на тему «Онтологічне представлення реєстру методичних ресурсів кафедри»

Виконав (-ла): студент (-ка) 4 курсу, групи ТМ-52

Бублик Анна Сергіївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викладач Колумбет В. П. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант Опис програмної реалізації ст. викладач Дацюк О. _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший бакалаврський

Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

Бублик Анні Сергіївні
(прізвище, ім’я, по батькові)

1. Тема роботи «Онтологічне представлення реєстру методичних ресурсів кафедри»

керівник роботи Колумбет Вадим Петрович старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”22” 05 2019р. № 1325С

2. Строк подання студентом роботи .06.2019

3. Вихідні дані до роботи програмний застосунок розроблено у середовищі Visual Studio 2015 на платформі .NET з використанням мови C#.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Створити онтологію реєстру методичних ресурсів кафедри, проаналізувати використання мов запитів для доступу до онтології. Розробити застосунок, що дасть змогу виконувати пошук по онтологічному джерелу без знання мови SPARQL користувачами.

5. Перелік ілюстративного матеріалу
«Мета роботи», «Актуальність теми», «Задачі, поставлені при виконанні», «Поняття онтології», «Створення онтології», «Дерево класів та зв'язки», «SPARQL-запит», «Взаємодія компонентів», «Функціональна схема системи», «Приклади роботи програми», «Висновки».

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Опис програмної реалізації	Дацюк О. А. ст. викладач		

7. Дата видачі завдання "14" жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	09.10.2019	
2.	Вивчення та аналіз задачі	14.10.2019-23.12.2019	
3.	Розробка архітектури та загальної структури системи	02.02.2019-03.03.2019	
4.	Розробка структур окремих підсистем	04.03.2019-14.04.2019	
5.	Програмна реалізація системи	15.04.2019-19.05.2019	
6.	Оформлення пояснювальної записки	20.05.2019-05.06.2019	
7.	Захист програмного продукту	14.05.2019	
8.	Передзахист	28.05.2019	
9.	Захист	17.06.2019-21.06.2019	

Студент _____ Бублик А. С.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Колумбет В. П.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення онтології реєстру методичних ресурсів кафедри та розробка програмного забезпечення генерації та виконання запитів до онтологічних джерел.

Об’єктом дослідження є онтологія та мова SPARQL. Було виконано огляд існуючих програмних засобів для створення онтології та зроблено висновки щодо їх переваг та недоліків. Створено клієнтський застосунок для виконання SPARQL-запитів. Продукт можна використовувати для пошуку даних в онтологіях. Було подано тези на конференцію «Наукові відкриття та фундаментальні наукові дослідження: світовий досвід» стосовно даної роботи.

Загальний обсяг роботи: 54 сторінки, 29 ілюстрацій та 25 бібліографічних найменувань.

Ключові слова: SPARQL, dotNetRdf, OWL, RDF, онтологія.

ABSTRACT

The purpose of the thesis is to create the ontology of the register of methodological resources of the department and develop software for generation and execution of requests for ontological sources.

The object of the study is ontology and language SPARQL. An overview of existing software tools for ontology was made and conclusions were drawn about their advantages and disadvantages. A client application has been created to execute SPARQL requests. The product can be used to search data in ontologies. The thesis was presented at the conference "Scientific discoveries and fundamental research: world experience" in relation to this work.

Total volume of work: 54 pages, 29 illustrations and 25 bibliographic titles.

Keywords: SPARQL, dotNetRdf, OWL, RDF, ontology.

ЗМІСТ

Перелік скорочень, умовних позначень і термінів 7

Вступ 8

1 Постановка задачі 10

2 Аналіз онтологічного представлення реєстру методичних ресурсів кафедри . 12

2.1 Поняття реєстру методичних ресурсів кафедри 12

2.2 Поняття онтології 14

2.3 Мова опису онтологій RDF 16

2.4 Мова опису словника RDFS 18

2.5 Мова опису веб-онтологій OWL 19

2.6 Поняття SPARQL-запиту 20

2.7 Аналіз існуючих програмних систем 21

2.7.1 Програмний продукт OntoEdit 22

2.7.2 Графічний редактор OilEd 23

2.7.3 Інструмент WebOnto 24

2.7.4 Середовище OntoStudio 25

2.7.5 Система Apollo 26

2.8 Висновки до розділу 27

3 Засоби розробки програмного продукту 28

3.4 Онтологічна модель 28

3.5 Система Protégé для створення онтології 29

3.6 Середовище розробки Visual Studio 2015 30

3.7 Платформа .NET Framework 4.5.2 32

3.8 Висновки до розділу 32

4 Опис програмної реалізації 33
4.1 Створення онтології 34
4.2 Написання запитів до розробленої онтології 40
4.3 Створення клієнтського застосунку 43
4.4 Висновки до розділу 44
5 Робота користувача з програмною системою 45
5.1 Системні вимоги та інсталяція 45
5.2 Сценарій роботи користувача з системою 45
5.3 Висновки до розділу 50
Висновки 51
Список використаних джерел 52
Додаток 1 55
Додаток 2 57
Додаток 3 68
Додаток 4 76

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

1) Онтологія – детальний опис деякої проблемної області.
2) W3C – міжнародна організація, що розробляє стандарти для всесвітньої павутини.
3) XML – мова розмітки ієрархічних даних.
4) RDF – стандарт W3C опису онтологій.
5) RDFS – розширення словника RDF для формалізації відношень.
6) OWL (мова веб-онтологій) – розширення RDFS, що привносить нові елементи описової логіки.
7) SPARQL – стандартизована мова запитів до онтологій.
8) C# – мова програмування.
9) DotNetRdf – бібліотека для доступу та роботи з онтологією.

ВСТУП

Сьогодні піднесення інформаційних технологій приводить до зростання обсягів інформації. Відповідно виникають задачі, пов’язані з інтеграцією корпоративної інформації, отриманою з різних джерел та у різних форматах, з метою зручного та швидкого пошуку та подання користувачеві.

З розвитком наукоємних сфер людської функціональності в сучасному світі зростає попит на комп'ютерні технології. Зростання обсягу інформації змушує знаходити нові способи її зберігання, уявлення, формалізації і систематизації, а також автоматичної обробки. Таким чином, зростає інтерес до всеосяжних баз знань, які можливо використовувати для різних практичних цілей. Великої уваги заслуговують системи, здатні без участі людини добути будь-яку інформацію з тексту. Як результат, розвиваються нові технології, покликані вирішити заявлені проблеми.

Актуальність обраної теми визначається конкретним спектром невирішених питань, що виникають з піднесенням інформаційних технологій та збільшенням обсягів інформативних ресурсів.

Одним з таких рішень є онтологія. Онтологія використовується для обробки семантичних зв’язків між сутностями предметної області. Крім того, онтологія разом з безліччю індивідуальних примірників класів складають базу знань. Найбільш перспективним та практичним способом опису онтології є візуальний підхід, хоча існують й інші мови та системи для опису. Такий підхід надає змогу безпосередньо візуально створювати онтології, що допомагає наочно розкрити і пояснити природу і композицію явищ. Візуальні моделі, наприклад, графи мають особливу когнітивну силу, фактично представляючи ресурси когнітивної графіки для структурування інформації. Існує безліч програмних пакетів, що можуть слугувати як первинний інструмент побудови онтологій.

Основною метою дипломної роботи є створення онтології реєстру методичних ресурсів кафедри та розробка програмного забезпечення генерації та виконання

запитів до онтологічних джерел, що було б доступним та легким у використанні для всіх користувачів і не потребувало б специфічних знань проектування запитів мовою SPARQL.

Для здійснення поставленої мети були висунуті такі завдання:

– проаналізувати методичні ресурси кафедри;
– визначити складові методичних ресурсів;
– описати систему зберігання ресурсів на кафедрі;
– структурувати методичні ресурси;
– створити онтологічну модель та оцінити її доцільність;
– розробити програмний продукт для роботи з онтологічними джерелами.

1 ПОСТАНОВКА ЗАДАЧІ

За роки існування кафедри кількість документів, методичних матеріалів та інших ресурсів зростає. Розвиток технологій не обійшов і нашу кафедру, саме тому матеріали містяться не лише в паперовому вигляді, а й в електронному. Для їх зберігання створюються бази даних, але вони достатньо великі. І якщо при розробці бази даних не було правильно передбачено її майбутній розвиток – то через декілька років пошук та обробка даних стає досить складною. Знаходити дані у великих базах даних стає все важче, бо з року в рік цієї інформації стає все більше.

Саме тому метою цієї дипломної роботи є організація, класифікація і керування цією великою кількістю даних. Крім того, стоїть задача по оптимізації витрати часу персоналу на пошук та обробку певних даних.

Для обробки інформації на кафедрі в дипломній роботі пропонується використати онтологію реєстру методичних ресурсів кафедри. Це певна структура, яка містить основні класи методичних ресурсів кафедри, їх взаємозв’язки, метадані та типи даних.

Дана задача включає в себе:

– аналіз предметної області;
– створення онтології кафедри;
– написання запитів до онтології;
– можливість виводити інформацію у зручному вигляді;
– можливість переглядати документи.

До компонентів розробленої системи входить:

– інтерфейс Protégé;
– запити мовою SPARQL;
– додаток написаний мовою C#;
– бібліотека dotNetRDF для доступу до онтології з додатку.

Важливо також виділити основні задачі програмного продукту та визначити, в яких цілях він буде використовуватися користувачами. Дану систему можна використовувати:

– для дослідження структури онтології;
– для отримання даних з онтології;
– для зручного та швидкого пошуку;
– для перегляду методичних ресурсів на кафедрі;
– для оцінки насиченості дисципліни методичними ресурсами.

Основною вимогою до розроблюваної системи є виконання SPARQL-запитів без знання користувачами мови. Тобто створення такого програмного продукту, де запити представлені в доступній формі для будь-якого користувача.

2 АНАЛІЗ ОНТОЛОГІЧНОГО ПРЕДСТАВЛЕННЯ РЕЄСТРУ МЕТОДИЧНИХ РЕСУРСІВ КАФЕДРИ

Спираючись на сукупність всіх перерахованих факторів у минулому розділі, виникає необхідність у реєстрі інформаційних ресурсів. Сьогодні у якості сучасної моделі, яка використовується для структурування інформаційних потоків використовують онтологію.

2.1 Поняття реєстру методичних ресурсів кафедри

Реєстр інформаційних ресурсів – це інформаційна система, яка проводить реєстрацію, обліку, накопичення і зберігання відомостей про інформаційні ресурси та надання доступу до цих ресурсів користувачам системи [1]. Простіше кажучи, реєстр є каталогом у якому зберігається опис, розміщення, умови доступу, інформація про власника, формат, права на коригування ресурсів та ін.

Інформаційні ресурси кафедри – систематизована інформація, що є доступною за допомогою інформаційних технологій, право на володіння, використання або розпорядження якою належить кафедрі, а також інформація, створення якої передбачено законодавством та яка обробляється фізичними особами відповідно до наданих їм повноважень.

Інформаційні ресурси кафедри – це результати інтелектуальної та практичної діяльності, що сформовані в усіх сфері роботи кафедри, зафіксовані і систематизовані на відповідних матеріальних носіях інформації, як окремі документи і масиви документів, банки і бази даних та знань, усі види архівів і бібліотек, музейні фонди, інформаційні ресурси які обробляються й передаються в інформаційних системах загального призначення, інші ресурси, що містять дані, відомості і знання які є об’єктом права власності кафедри незалежно від форми власності на час їх створення і мають споживчу цінність, а також такі, що

призначені для розвитку і задоволення потреб студентів та викладачів та підлягають захисту згідно з визначеною політикою безпеки й чинним законодавством.

Створення реєстру методичних ресурсів кафедри вимагає неперервного системного аналізу інформаційних ресурсів, їх взаємозв'язків та захищеності. При цьому реєстр слід розглядати як розподілену складну інформаційну систему, оскільки він наділений усіма властивостями складних систем, а саме: великою кількістю складових компонент, взаємодією з навколишнім середовищем, ієрархічною структурою та мінливістю у часі.

Реєстр має забезпечувати облік інформаційних ресурсів, згрупованих за певною спільною ознакою. Наприклад: за формою власності – реєстр інформаційних ресурсів кафедри; за тематикою – галузеві стандарти тощо.

До інформаційних ресурсів кафедри можна віднести всю належну інформацію включаючи окремі документи і масиви документів, які належать до наукового, навчально-методичного та організаційного напрямків роботи кафедри.

До методичних ресурсів кафедри відноситься величезний пласт документів різних напрямків та різних форматів, пов'язаних з організацією навчального процесу [2]:

– навчальних планів; робочих навчальних планів;

– програм навчальних дисциплін, робочих програм кредитних модулів, плани практичних і семінарських занять, критерії оцінювання рейтингу студентів;

– підручники, навчальні посібники, конспекти лекцій;

– навчальні та методичні посібники до лабораторних робіт, курсових проектів,

– варіанти індивідуальних семестрових завдань, теми курсових проектів/робіт

– засоби діагностики для поточного та семестрового контролю результатів навчання та критеріїв оцінювання;

– завдання для проведення комплексних контрольних робіт з навчальних дисциплін та критеріїв оцінювання рівня підготовки студентів для проведення акредитації освітньої програми, моніторингу залишкових знань і вмінь;

– навчально-методичні матеріали дистанційного навчання (автоматизовані

навчальні комплекси: відео-лекції, електронні підручники та практикуми, віртуальні лабораторні роботи, засоби тестового поточного контролю; методичні рекомендації щодо особливостей організації дистанційного і змішаного навчання тощо).

2.2 Поняття онтології

Сьогодні у якості сучасної моделі, яка використовується для структурування інформаційних потоків використовують онтологію. Онтологія дозволяє побудувати концептуальну модель предметної області, встановити різні типи зв'язків між класами та даними предметної області.

Поняття онтологія зараз активно застосовується в інформатиці та штучному інтелекті. Цей термін прийшов з філософії, де позначав частину метафізики – вчення про усьому сущому, про його найбільш загальних філософських категоріях, таких як буття, субстанція, причина, дія, явище. При цьому онтологія як наука претендувала на повне пояснення причин всіх явищ.

За іншим визначенням, онтологія – це загальнодоступна прийнята концептуалізація деякої області знань, що може послужити основою, фундаментом для її моделювання та визначає шляхи взаємодії між системами, що використовують знання з цієї області, і обов'язково включає домовленості щодо представлення теоретичних основ області знань [3].

В інженерії знань під онтологією мається на увазі детальний опис деякої проблемної області, який застовується для формального і декларативного визначення її концептуалізації.

Кажучи простими словами онтологія – це точна конкретизація деякої області, що складається із словника термінів цієї галузі та безлічі логічних зв'язків (типу «елемент-клас», «частина-ціле»), які описують співвідношення цих термінів між собою.

По відношенню до комп’ютерних наук поняття «онтологія» вперше згадується в роботі Томаса Грубера, який вирішував задачу створення механізму взаємодії системи баз знань один з одним.

Принципи проектування і реалізації онтологій, запропонованих Грубером:

– ясність – всі терміни повинні бути зрозумілі, а їх зміст точно переданий;
– узгодженість - не повинно виникати ніяких протиріч, всі терміни повинні однозначно інтерпретуватися;
– можливість розширення – онтологія повинна дозволяти доповнювати і конкретизувати вже існуючі поняття без їх зміни;
– мінімальна залежність від кодування – концепти можуть бути реалізовані в різних системах знань, залежні тільки від рівня уявлення, а не від кодування;
– мінімум онтологічних зобов'язань – онтологія повинна описувати найбільш суттєві поняття предметної області, щоб залишалося можливість її розширення і спеціалізації.

Структура онтології налічує чотири основні категорії:

– поняття;
– відносини;
– аксіоми;
– екземпляри.

Поняття являє собою визначення та опис якоїсь конкретної сутності або явища, наприклад, тварина, будівля. В центрі будь-якої онтології є класи (або поняття), які описують поняття. Характеристикою класів є їх метадані. Так, наприклад, методичні ресурси можна поділити на два глобальні класи: нормативні документи та навчальні матеріали. До їх метаданих можна віднести автора, спеціальність, курс, дисципліну, дату публікації та інші.

Поняття пов’язуються відносинами, що дає змогу пов’язати класи і тим самим описати їх. Найбільш поширеним типом відносин є ставлення категоризації, тобто належність до певної категорії. Цей тип відносин має ряд інших назв [4], що зустрічається в різних дослідженнях:

– таксономічне відношення;
– ставлення IS-A;
– клас – підклас;
– гіпонім – гіперонім;
– родове відношення;
– відношення a-kind-of.

Класи поєднані між собою різного роду відносинами, які об’єднують воєдино сутності і описують їх. Найпоширенішим типом відносин, що обов’язково наявне у всіх онтологіях, є відношення класифікації (категоризації), тобто віднесення до певної категорії. Цей тип відносин має ряд інших назв , що зустрічається в різних дослідженнях [5]:

– ставлення «Є»;
– клас, підклас;
– таксономічне відношення;
– відношення «Один-з».

Аксіоми використовуються, щоб записати висловлювання, які завжди правдиві [6]. Аксіоми використовуються для визначення умов відповідності між категоріями та відносинами. Іншими словами, аксіоми – очевидні твердження, що зв’язують поняття і відносини. Вони дозволяють висловити ту інформацію, яка не може бути відображена в онтології за допомогою побудови ієрархії понять і установки різних відносин між поняттями. Прикладом аксіоми може бути наступне висловлювання: «Якщо X голодний, то X коли-небудь поїсть».

Екземпляри – це окремі представники класу сутностей або явищ, тобто конкретні елементи будь-якої категорії [7].

2.3 Мова опису онтологій RDF

Модель для опису ресурсів (RDF) – це універсальна, базова система для кодифікації, обміну і повторного використання структурованих метаданих. Вона підтримує сумісність веб-додатків, які здійснюють обмін інформацією, що

розуміють машини [8]. Атомарним об’єктом в RDF є триплет: суб’єкт – предикат – об’єкт. Вважається, що будь-який об’єкт, можна описати в термінах простих властивостей і значень цих властивостей. Набір таких триплетів називають RDF-графом.

Важливо відмітити, що RDF дає можливість використовувати таку форму подання, в якій будь-який вираз RDF в трійці може бути об’єктом або значенням, а це значить, що графи можуть бути вкладеними та лінійними. У Web це дозволяє нам, наприклад, оскаржити або підтримати вирази, створені іншими людьми.

Головною метою для RDF являється висунення базової моделі даних «об’єкт – атрибут – значення» для метаданих. Крім цієї передбачуваної семантики, описаної в стандарті, RDF не містить будь-яких чітких правил, орієнтованих на моделювання даних.

Властивості RDF можна вважати атрибутами ресурсів і в цьому сенсі вони відповідають традиційним парам виду «атрибут–значення». Властивості RDF відображають також відносини між ресурсами. Однак RDF не тільки не надає механізми для опису цих властивостей, а й механізми для опису відносин між цими властивостями і іншими ресурсами.

Модель даних RDF, що представляє оператори RDF в синтаксично нейтральної формі, дуже проста і базується на 3-х типах об'єктів: ресурсах, властивостях і операторах. Перші два однозначно ідентифікуються по URI.

Ресурси – це все, що можна описати виразом RDF: веб-сторінка або її частина, елемент XML всередині вихідного документа і навіть ціла сукупність веб-сторінок або об’єкт, до якого немає прямого доступу по мережі.

Властивості – це конкретний аспект, характеристика, визначення або ставлення, що використовується для опису ресурсу. Кожна властивість має певний сенс. Воно визначає допустимі значення, типи ресурсів, які може описати, і зв'язку з іншими властивостями. Властивості, асоційовані з ресурсом, ідентифікуються по імені і мають значеннями.

Оператори – ресурс з певним властивістю, який визначається ім’ям, і значенням цієї властивості для певного ресурсу, формують оператор RDF. Таким

чином, оператор – це триплет, що містить в собі суб'єкт (ресурс), предикат (властивість) і об'єкт (значення). Об'єкт оператора (значення) може бути виразом (послідовністю символів або будь-яким іншим примітивним типом, описуваних XML) або іншим ресурсом.

Існують три типи вузлів в RDF-графі: унікальні ідентифікатори, літерали та пусті вузли.

Онтологічна структура RDF є триплетом $< R, P, h >$ [9], де:

– R – виключна множина унікальних УІР, що однозначно ідентифікує всі ресурси, літерали та пусті вузли;
– P – множина властивостей;
– h – відображення P на множину $R \times R$, тобто операція, що пов'язує кожну з властивостей множини P до її домену (власників) та можливих значень.

2.4 Мова опису словника RDFS

Так як RDF не робить жодних припущень про предметну область, не задає семантику, для досягнення цього була розроблена RDFS (англ. RDF Schema) — мова для задання семантики певної області знань. Фактично RDFS є розширенням базового словника RDF [10].

Мова RDFS всього на всього розширює мову RDF. Вона надає механізми для опису груп пов'язаних ресурсів і відносин між цими ресурсами. Всі визначення RDFS виражені на RDF (тому RDF і називається «самоописуючим»). Нові терміни, що вводяться RDFS, такі як «домен», «діапазон», «властивості» є ресурсами RDF.

RDFS – це стандартна лексика, як RDF. Якщо в словнику RDF є терміни, які допоможуть дати базове визначення / опис примірників, в словнику RDFS є терміни, що дають змогу описувати класи. Наприклад, є визначення терміна rdfs: subClassOf. За допомогою цього терміна можна описати той свідчення, що клас є підкласом іншого.

Існує перелік аксіоматичних тверджень, які завжди справедливі в семантиці RDFS. Наприклад:

— за типом <rdf:type rdf:type rdf:Property>, <rdf:subject rdf:type rdf:Property>;

— за областю застосування <rdf:type rdfs:domain rdfs:Resource>, <rdfs:domain rdfs:domain rdf:Property>;

— за областю значень (rdf:type rdfs:range rdfs:Class).

На жаль, RDFS не є достатньо виразною мовою. Тому неможливо задати обмеження на відношення. Для реальних програмних продуктів було потрібне щось краще. Саме тому виникла мова OWL (англ. Web Ontology Language). Мова OWL заснована на описовій логіці. З нею стало можливим представити область знань в формі онтології. На рисунку 2.1 наведено приклад опису підкласу.

```
PREFIX rdfs:<https://www.w3.org/2000/01/rdf-schema#>
<http://yourdomain.com/Teacher> rdfs:subClassOf <http://yourdomain.com/Person>
```

Рисунок 2.1 – Приклад опису підкласу мовою RDFS

Отже, у RDF є умови для створення екземплярів, а у RDFS є умови для створення класів. Застосовуючи обидва ці варіанти, можна зробити більш докладний опис даних.

2.5 Мова опису веб-онтологій OWL

Мова OWL (Web Ontology Language) являє собою мову, призначену для опису онтологій. OWL побудована як розширення RDF і RDFS. Це означає, що основна конструкція – це трійка мови RDF. У цьому контексті мову OWL можна розглядати як розширений варіант RDFS, що дозволяє не тільки описувати класи і властивості, але також задавати обмеження на їх використання. Мовою дискрипційної логіки це означає, що логіка, що лежить в основі OWL, містить крім опису відносин також і аксіоми, що задають співвідношення між даними відносинами і різного роду обмеження останніх [11].

OWL-класи можуть бути вказані як логічні комбінації (об’єднання, доповнення, перетин) інших класів або як перечислення деяких об’єктів. Області застосування OWL-властивостей є OWL-класами, як і області значень. З’явилась

можливість зазначити властивість як транзитивну, симетричну, функціональну чи інвертовану іншого відношення [12].

Існує декілька варіацій мови OWL:

– мова OWL Lite дозволяє будувати класифікацію понять і деякі прості умови їх погодження;

– мова OWL DL (Description Logic) орієнтований на системи опису знань і системи логічного програмування, надає користувачам можливості мови зі збереженням формально логічних заключень і можливості розв'язання (всі обчислення виконуються за кінцевий час);

– мова OWL Full надає всі можливості мови і вибір синтаксичних засобів, але не є неминучим зберігання при цьому обчислювальної повноти і можливість розв'язання.

2.6 Поняття SPARQL-запиту

Для запитів до даних, представлених у RDF, розроблена мова SPARQL, яка є рекомендацією консорціуму Всесвітньої павутини (W3C) [13]. Її назва – рекурсивний акронім, що розшифровується як SPARQL Protocol And RDF Query Language. SPARQL — протокол та мова запитів до RDF. Користуючись звичайним протоколом та мовою SPARQL, прикладна комп’ютерна програма може розібрати RDF-описи ресурсів та отримувати з мережі потрібну інформацію.

Більша частина запитів SPARQL містить набір шаблонів триплетів, що є основним графовим шаблоном. В результаті запиту отримуємо послідовність рішень із даними, які відповідають графовому шаблону запиту. Запит може визначати отримання одного або множини рішень, або взагалі відсутність рішень.

Специфікація описує чотири типи запитів:

– тип SELECT — повертає змінні та їхні значення;

– тип CONSTRUCT — дозволяє отримати інформацію з точки доступу SPARQL в форматі RDF;

– тип ASK — повертає результат істина/хибність;

– тип DESCRIBE — повертає опис RDF-ресурсу.

Нижче наведений приклад демонструє SPARQL-запит, що дозволяє визначити назву книги із заданого графа даних (рисунок 2.2). Запит складається з двох частин: умова SELECT задає змінні, які повинні відображатися в результатах запиту, а умова WHERE надає основний графовий шаблон, якому повинні відповідати графові дані. У цьому прикладі основний графовий шаблон складається з одного шаблону триплета з єдиною змінною (? Title) на місці об'єкту.

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Рисунок 2.2 – Приклад простого SPARQL-запиту

Цей запит має єдине рішення, результати якого наведено нижче (рисунок 2.3).

title
"SPARQL Tutorial"

Рисунок 2.3 – Результат SPARQL-запиту.

Служба SPARQL-endpoint — це служба сумісна з SPARQL. Вона дозволяє користувачу отримати бажані дані з бази знань шляхом виконання запитів [14]. Після обробки запиту результат повертається в одному з визначених форматів. Таким чином, SPARQL-точки доступу являє собою сервіс, що забезпечує стандартний інтерфейс до бази знань.

2.7 Аналіз існуючих програмних систем

Опис онтології доволі складний процес, який вимагає знання мови OWL. Людині без знань цієї мови неможливо побудувати та описати онтологію. Саме тому, задля полегшення цієї задачі та зменшення обсягу необхідних знань при її створені, було розроблено ряд програмних продуктів з графічним інтерфейсом. За

допомогою таких програм можна не лише легко описати онтологію, а й побудувати до неї запити.

2.7.1 Програмний продукт OntoEdit

Програмний продукт OntoEdit спочатку був розроблений в інституті AIFB (Institute of Applied Informatics and Formal Description Methods) Університету Karlsruhe [15]. Зображення інтерфейсу користувача наведено на рисунку 2.4.

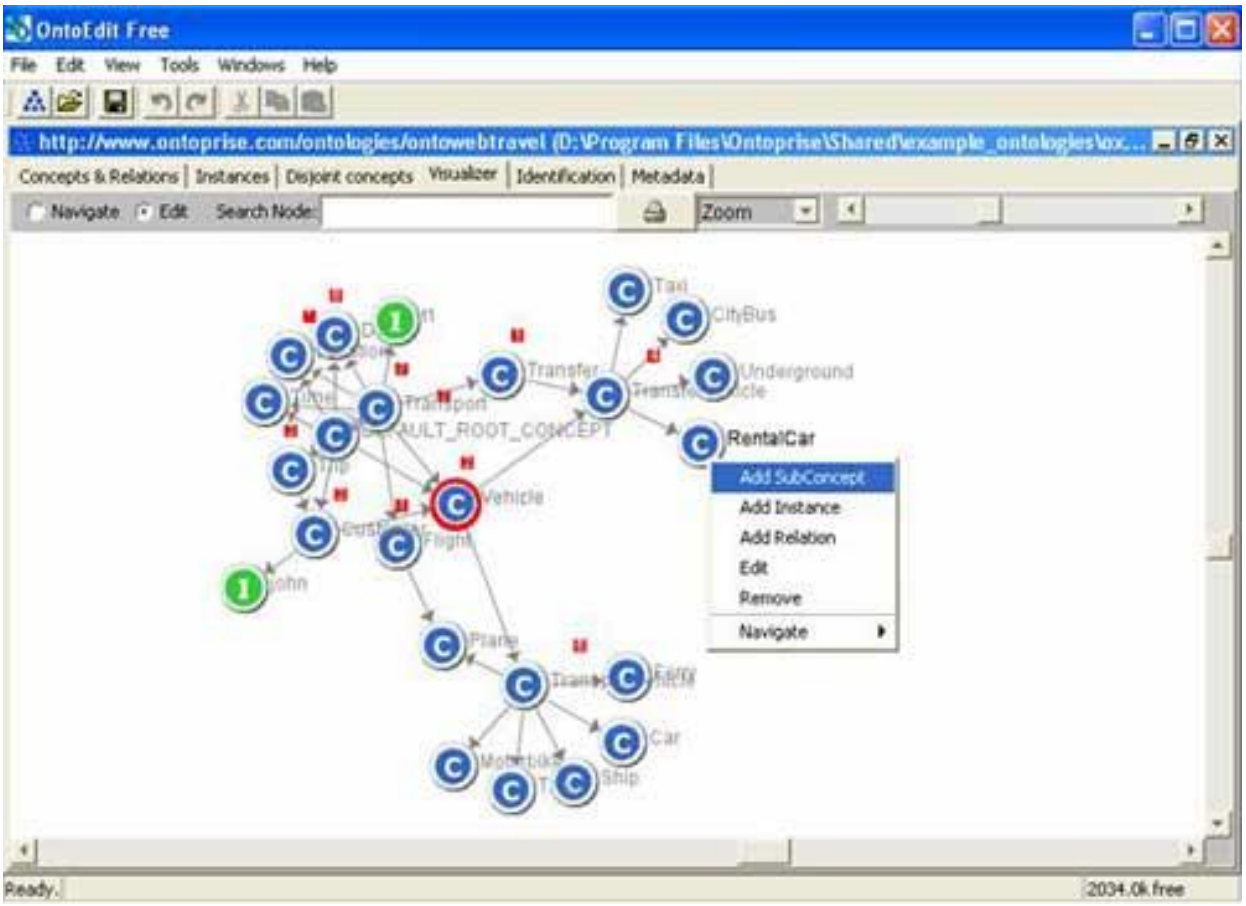


Рисунок 2.4 – Інтерефейс OntoEdit

Програма надає змогу перевіряти, переглядати, кодувати та модифікувати онтологію. OntoEdit – автономний Java-додаток, який можна локально встановити на комп'ютері, але його коди закриті. Архітектура OntoEdit подібна Protégé.

Основними перевагами інструменту є: зручність та легкість використання; можливість розробки онтології під керівництвом методології і за допомогою

процесу логічного висновку; розробка аксіом; можливість розширення структури за допомогою плагінів, а також наявність дуже хорошої документації.

2.7.2 Графічний редактор OilEd

Редактор OilEd – автономний графічний редактор онтологій, розроблений в Манчестерському університеті в рамках європейського IST проекту On-To-Knowledge [16]. Має зрозумілий і інтуїтивний стиль інтерфейсу користувача і переваги підтримки міркування (виявлення логічно суперечливих класів і прихованих відносин підкласу).

Зображення інтерфейсу користувача наведено на рисунку 2.5.

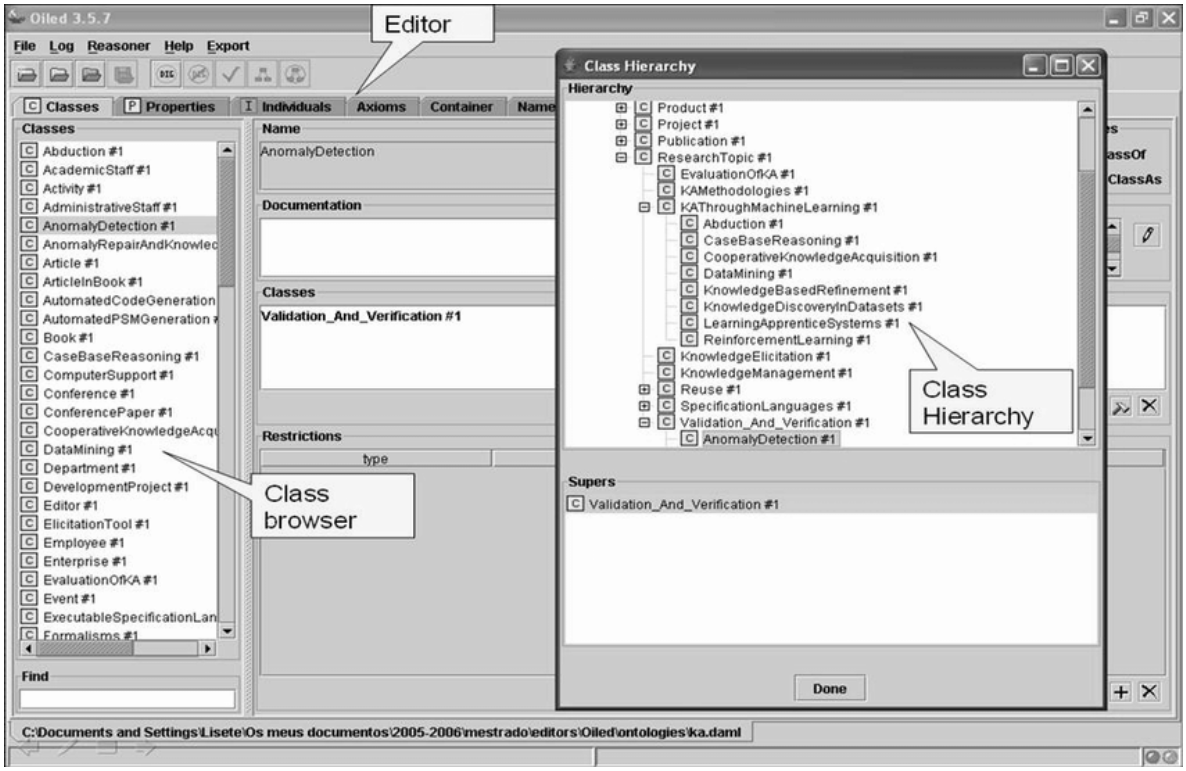


Рисунок 2.5 – Інтерефейс OilEd

У даний час спостерігається зростання популярності редактора OilEd. Він використовується як для навчання, так і для дослідження. Інструмент вільно поширюється по загальнодоступній ліцензії GPL.

З недоліків можна виділити відсутність підтримки примірників. Існуюча версія не забезпечує повне середовище розробки – не підтримується розробка онтологій великого масштабу, міграція та інтеграція онтологій, контроль версій і т.д. OilEd можна розглядати як "NotePad" редакторів онтологій, що пропонує достатню функціональність, щоб дозволити користувачам будувати онтології і продемонструвати, як можна використовувати механізм міркування FaCT для перевірки онтології на несуперечливість.

2.7.3 Інструмент WebOnto

Інструмент WebOnto розроблено для Tadzebao – інструменту дослідження онтологій і призначений для підтримки спільного перегляду, створення і редагування онтологій [17]. Його цілі – простота використання, надання коштів масштабування для побудови великих онтологій.

Зображення інтерфейсу користувача наведено на рисунку 2.6.

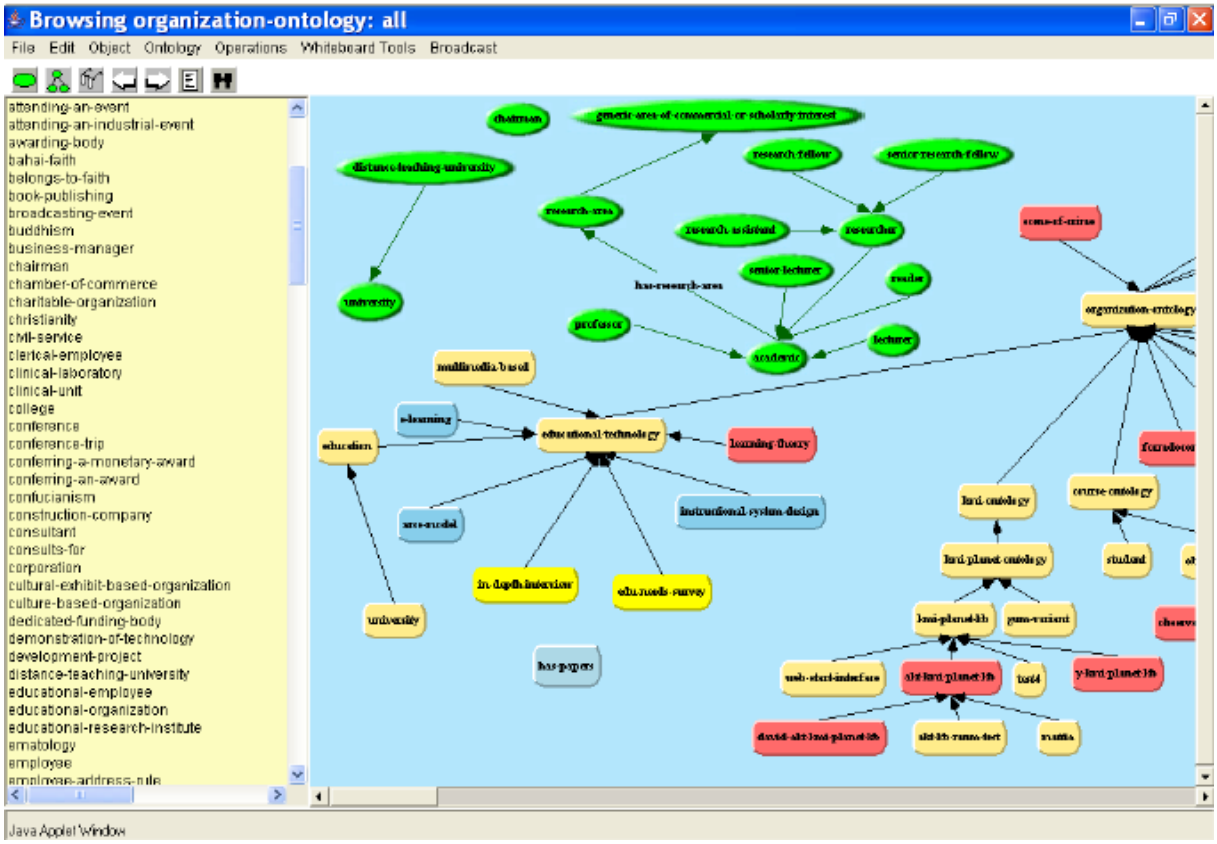


Рисунок 2.6 – Інтерефейс WebOnto

Інструмент має ряд вигідних особливостей: збереження структурних діаграм, відокремлений перегляд відносин, класів, правил і т.д. Інші можливості включають спільну роботу декількох користувачів над онтологією, використання діаграм, функцій передачі і прийому і ін.

2.7.4 Середовище OntoStudio

Середовище OntoStudio є найбільш широко використовуваним комерційним середовищем моделювання для створення та підтримки онтологій. Вона характеризується комплексними функціями для моделювання інтуїтивної онтології та різноманітними варіантами імпорту загальних структур, схем і моделей. Серед основних функцій – інструмент Mapping, який також дозволяє бізнес-користувачам моделювати складні залежності або інтегроване тестове середовище, що забезпечує гарантію якості в будь-який час [18]. Зображення інтерфейсу користувача наведено на рисунку 2.7.

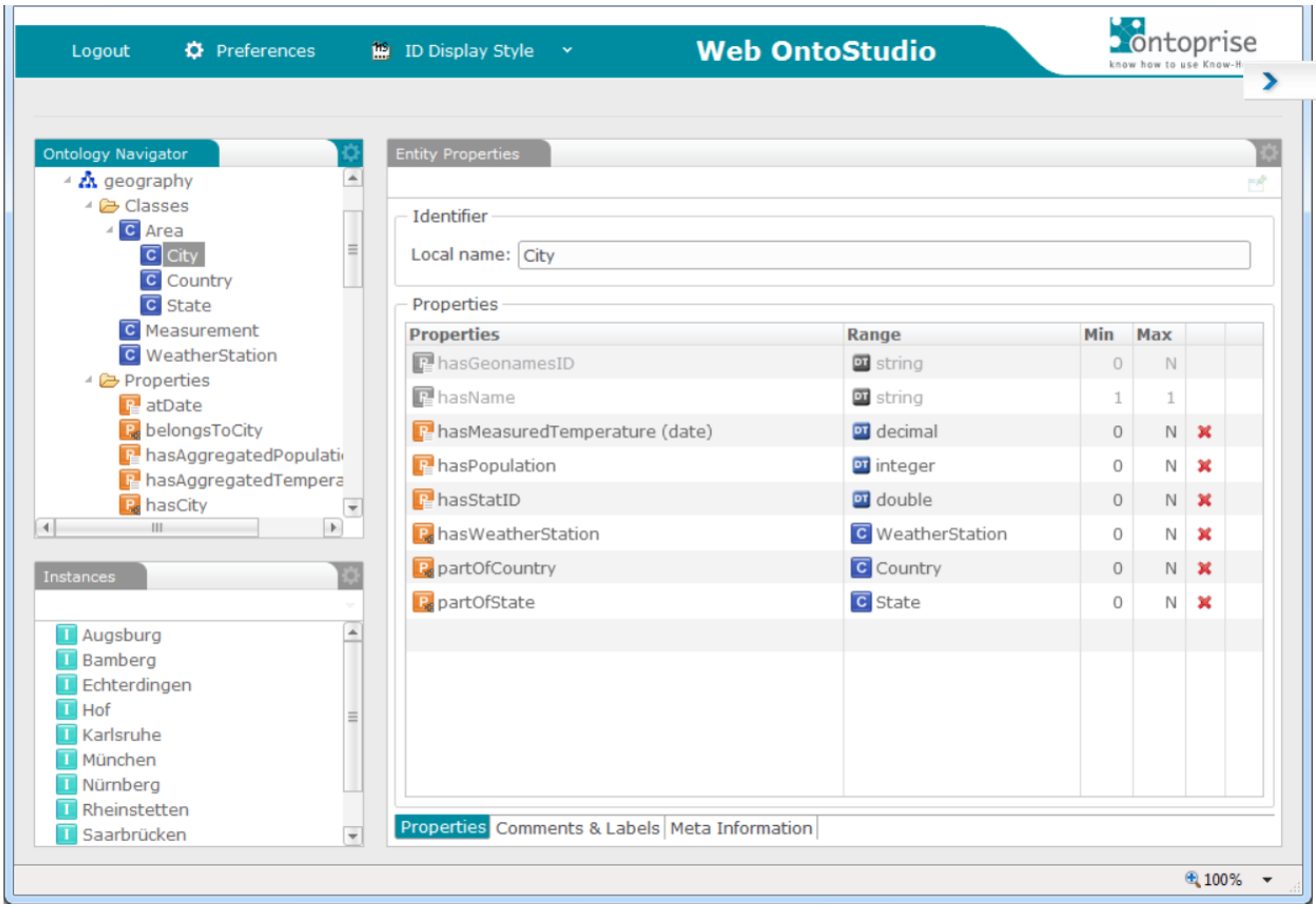


Рисунок 2.7 – Інтерефейс OntoStudio

Використовуючи OntoStudio, декілька редакторів можуть одночасно створювати та розширювати онтології, використовуючи сервер співпраці. Створені запити можна експортувати як веб-службу та інтегрувати в будь-яку програму.

2.7.5 Система Apollo

Apollo – зручний додаток для моделювання знань. Моделювання засноване на основних примітивах, таких як класи, екземпляри, функції, відносини і т. п. База знань складається з онтології, які ієрархічно організовані. Інтерфейс системи наведено на рисунку 2.8.

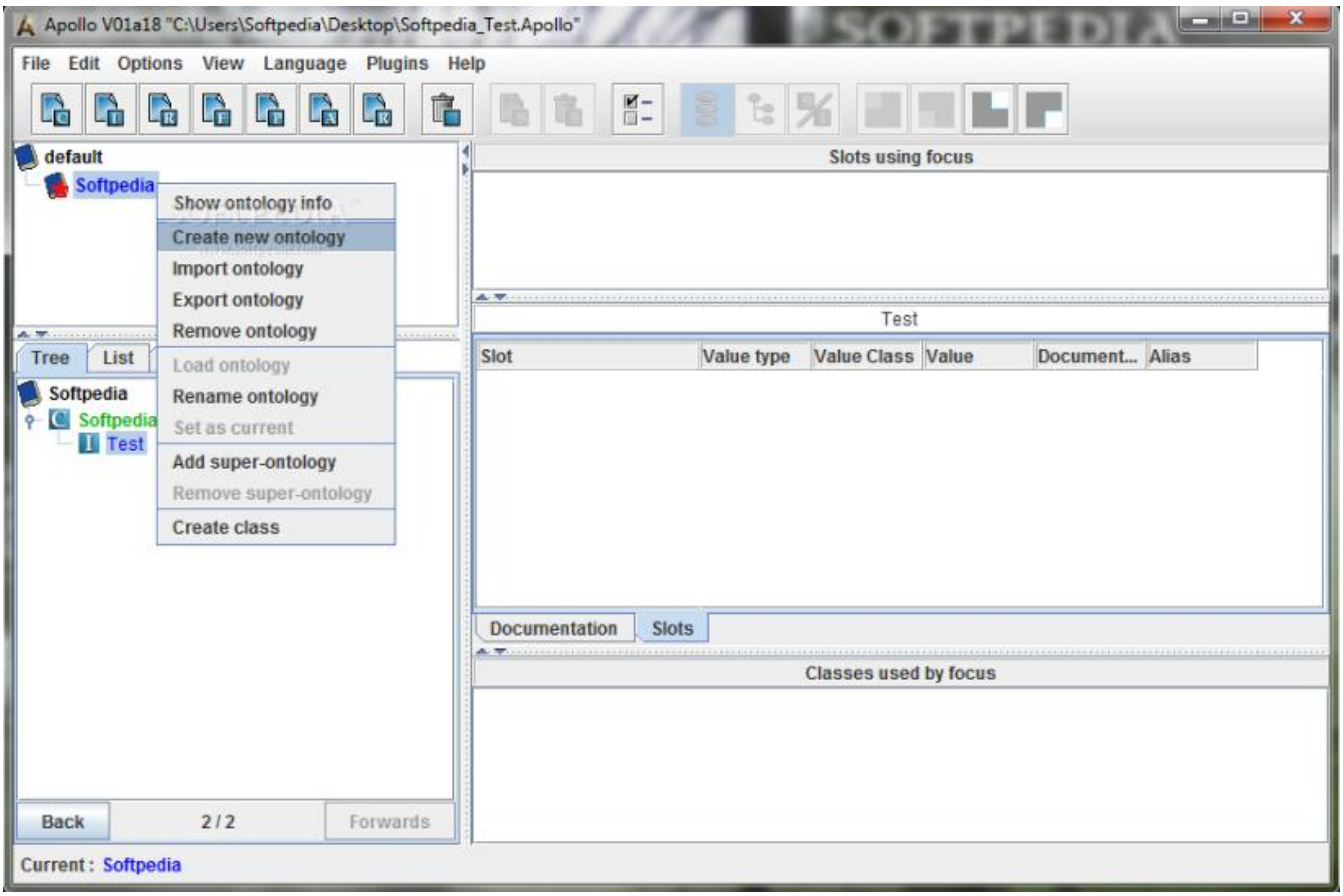


Рисунок 2.8 – Система Apollo

Онтологія може успадковувати іншу онтологію, а потім застосовувати класи успадкованої онтології як свої власні. Кожна онтологія успадковує принаймні одну онтологію – онтологію за замовчуванням, що містить всі примітивні класи: логічне, цілочисельне, float, string, list і т. п. Клас містить слоти двох типів: без шаблонів і

слотів шаблонів. Apollo в даний час не підтримує слоти класу без шаблонів. Для кожного класу можна створити кілька примірників. Примірник успадковує всі слоти класу. Кожен слот має набір фасетів. [19]

2.8 Висновки до розділу

У розділі виділено необхідні об’єкти інформаційних ресурсів кафедри. Крім того, наведено огляд редакторів онтології та обґрунтовано вибір редактора Protege для роботи з онтологією.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для розробки програмного продукту було використано наступні засоби:

– середовище Visual Studio 2015;

– система Protégé для створення онтології.

Перш за все для доступу до методичних ресурсів необхідно створити онтологію. Саме для цього і використовується Protégé.

За допомогою середовища Visual Studio 2015 мовою C# та з використанням бібліотеки dotNetRDF було спроектовано та розроблено десктопний додаток, що дає змогу переглядати методичні ресурси кафедри зручним для користувача способом.

3.4 Онтологічна модель

У наш час для створення і підтримки онтологій існує безліч різноманітних інструментів. Кожен з інструментів оснащений різними функціями, найбільш поширені з них: створення онтології, редагування та перегляд, підтримка документування, імпорт і експорт онтологій різних форматів і мов, підтримка графічного представлення онтологій, можливість управління бібліотеками онтологій і ін.

Найбільш відомими інструментами для роботи з онтологією є:

– система WebOnto

– середовище WebODE

– середовище Protégé

– система Ontolingua

– середовище OntoEdit

– графічний редактор OilEd

– система OntoSaurus

– інструмент OpenSemanticFramework

При виконані дипломної роботи для розробки онтології було використано середовище розробки Protege версії 5.5.0.

За допомогою середовища Protege було створено онтологію реєстру методичних ресурсів кафедри АПЕПС. За допомогою середовища проведено структуризацію інформації, що надає можливість швидшого пошуку необхідного ресурсу. Крім того, за допомогою графу можна переглянути структуру онтології, що є більш зручним та наглядним способом представлення.

Основним завданням розробки і створення онтології є:

– визначення основних класів і підкласів, побудова ієрархії класів;
– призначення атрибутів класів і властивостей екземплярів;
– заповнення онтології екземплярами;
– написання базових запитів SPARQL.

3.5 Система Protégé для створення онтології

Система Protégé– це не експертна система, а також не є програмою, яка безпосередньо будує експертні системи; натомість Protégé – це інструмент, що допомагає розробникам експертних систем створювати свої інструменти, спеціально пристосовані для набуття знань в конкретних прикладних областях [20].

Protégé – це безкоштовна платформа з відкритим кодом, яка надає користувачам набір інструментів для побудови моделей доменів та програм, що базуються на знаннях, з онтологіями. Був розроблений Стенфордським центром досліджень біомедичної інформатики в Медичній школі Стенфордського університету. Остання версія програми - Protégé 5. Для компьютерів вона має версії під операційні системи Windows, Mac OSX та Linux [21].

Архітектура модуля Protégé може бути адаптована для побудови як простих, так і складних онтологічних додатків. Розробники можуть інтегрувати результати роботи Protégé з системами правил або іншими розв'язувачами задач для побудови широкого спектру інтелектуальних систем.

Існує безліч версій даного програмного продукту, кожна з яких оснащена новими функціями. Прикладами одних з перших є Protégé-II та Protégé-2000.

Система Protégé-II оснащена функціями необхідними для реалізації бази знань і засновує кроки, пов’язані з методами пошуку розв’язку задач.

У Protégé-2000 додано три покращення. По-перше, Protégé-2000 легко взаємодіє з іншими системами. По-друге, для покращення практичності і відповідності нової моделі знань, Protégé-2000 був спроектований як єдина цілісна прикладна програма. І по-третє, для забезпечення більшої гнучкості, спроектований Protégé-2000 базується на змінній архітектурі, що підтримується мовою програмування Java.

Окрім того, Protégé-2000 має новий інтерфейс, який спрощує користувачам роботу зі створення бази.

Мова Protege OWL дозволяє описувати не тільки поняття, а й конкретні об'єкти. Вона має широкий набір операторів. Наприклад, перетин, об'єднання і заперечення. Вона організована на логічній моделі, яка дозволяє створювати визначення, відповідно неформальному опису. Таким чином, формулювання складних понять можуть бути побудовані на основі визначень простіших понять. Крім того логічна модель дозволяє використовувати міркування, які можуть перевірити чи затвердження і визначення в онтології взаємно узгоджені і можуть також з'ясувати, які концепції відповідають заданим визначенням [22].

3.6 Середовище розробки Visual Studio 2015

Середовище Visual Studio 2015 – це інтегроване середовище розробки програмного забезпечення від компанії Microsoft. За допомогою Visual Studio можна створювати додатки для Windows, iOS, Android та інших платформ. У Visual Studio включені інструменти не тільки для створення desktop додатків, але і web, мобільні і хмарні інструменти розробки. Є можливість написання коду на таких мовах як: C ++, C #, Visual Basic, F #, JavaScript, Python, TypeScript [23]. Крім усього цього вона

включає в себе конструктори, редактори, відладчики, а також величезну кількість розширень для різних областей застосування - від PHP до ігор.

На рисунку 3.1 зображено інтерфейс Visual Studio 2015.

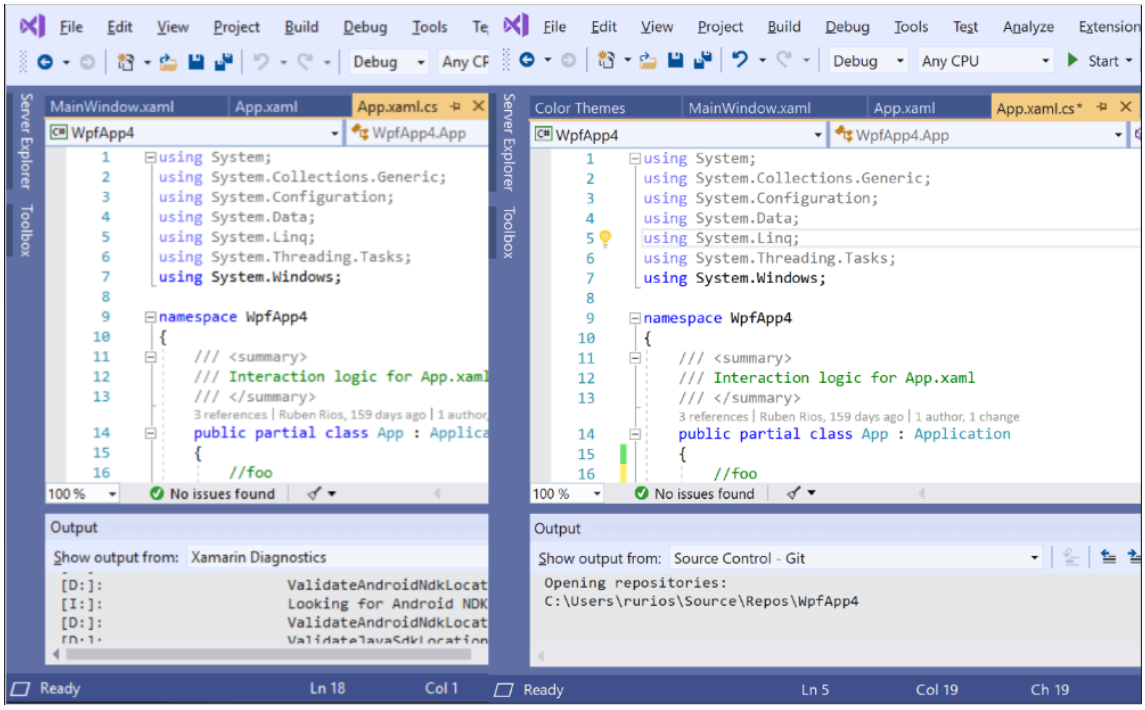


Рисунок 3.1 – інтерфейс Visual Studio 2015

Середовище Visual Studio 2015 року включає в себе наступні нові можливості і поновлення:

– випускається в трьох редакціях (раніше було чотири);

– крос-платформна підтримка мобільних пристроїв (Android, IOS, і Windows);

– покращення в C ++;

– зміни в налагодженні і діагностиці;

– платформа .NET Framework 4.6;

– покращена інтеграція Visual Studio і GitHub;

– і безліч інших.

3.7 Платформа .NET Framework 4.5.2

Платформа .NET Framework – це технологія, яка підтримує створення і виконання нового покоління додатків і веб-служб XML [24]. При розробці платформи .NET Framework були поставлені наступні цілі:

– оснащення платформи узгодженим об’єктно-орієнтованим середовищем програмування, що забезпечує збереження коду локально, дозволяє виконувати об’єктний код, який розміщено в мережі інтернет або для віддаленого доступу;

– створення середовища для виконання коду, яке оснащене системою контролю версій та зменшує проблеми при розгортанні програмного забезпечення;

– створення середовища для виконання коду, яке надає змогу безпечно виконувати код, включаючи код від невідомих постачальників;

– розробка середовища для виконання коду, яке поліпшує продуктивність виконання сценаріїв у середовищі та інтерпретування коду;

– встановлення єдиних парадигм розробки для різних видів клієнтських застосунків, наприклад, Windows і веб-додатки.

Саме за допомогою цієї платформи, написаний мною код конвертується в машинні операції, що дозволяє машині (ноутбуку) виконувати безперервну роботу розробленої мною програми.

3.8 Висновки до розділу

У розділі описана онтологічна модель та обґрунтовано вибір програмного забезпечення для створення онтології та розробки додатку.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення системи було використано два середовища: Protégé та Visual Studio 2015. Взаємодія компонентів системи зображено на рисунку 4.1.

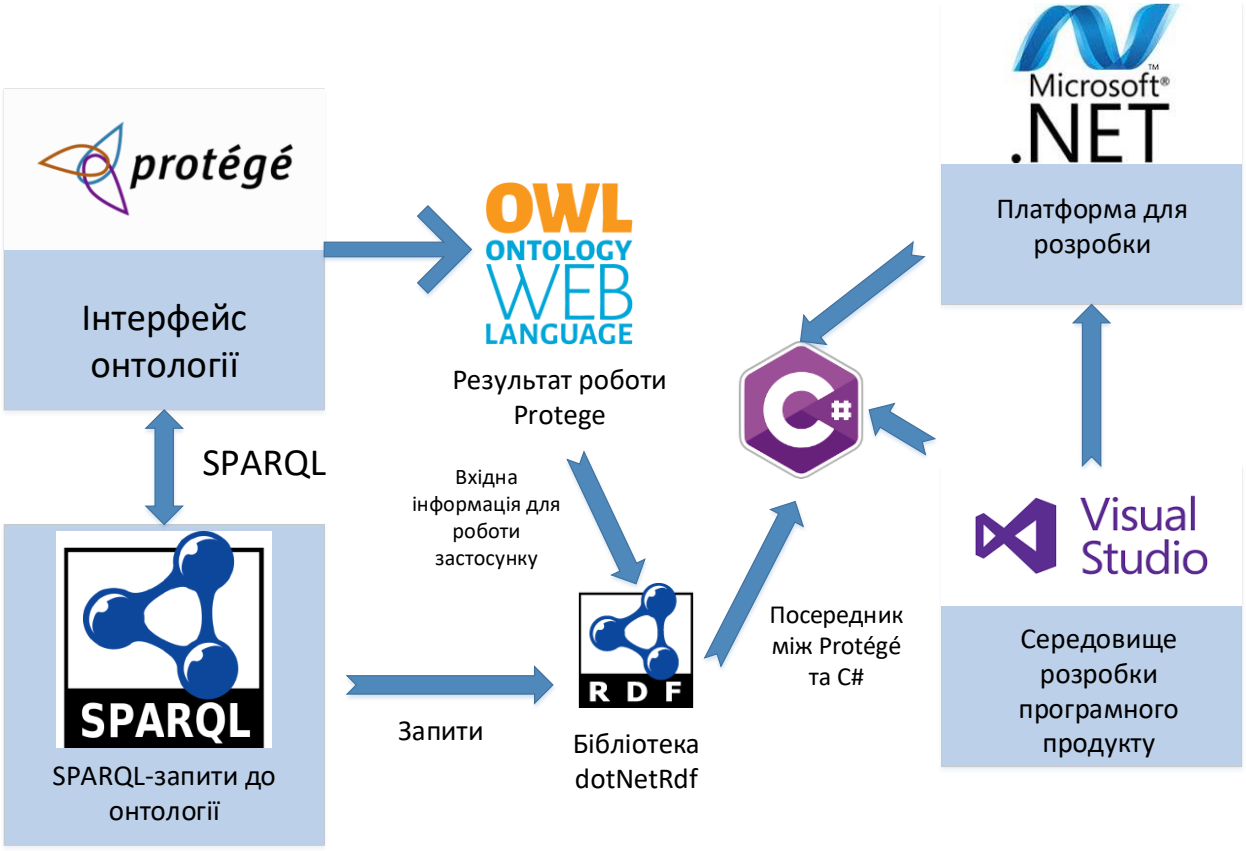


Рисунок 4.1 – Взаємодія компонентів системи

Більша частина дипломної роботи – створення онтології реєстру методичних ресурсів кафедри. Саме ця частина розроблялась в середовищі Protégé. Результатом роботи Protégé є файл формату owl. Саме цей файл і використовується при розробці програмного продукту. Клієнтський додаток написаний мовою C# на платформі .NET у середовищі Visual Studio 2015. Оскільки основним способом пошуку по онтології являється SPARQL-запит, то клієнтський застосунок базується на таких запитах. Єдиною проблемою при розробці застосунку є відсутність прямого доступу до owl-файлу та роботи з ним безпосередньо мовою C#. Задля вирішення цієї проблеми і використовується бібліотека dotNetRdf, яка виступає посередником між Protégé та C#.

Як відомо, кожна система виконує якісь конкретні функції. Функціонал системи наведено на рисунку 4.2.

Рисунок 4.2 – Функціонал системи

Оскільки система складається з онтології та клієнтського застосунку, то повинно бути два актори. Один з них – адміністратор, яким розуміється в понятті «онтологія» та може виконувати редагування онтології. Під редагуванням онтології розуміється додавання нових класів, створення нових зв’язків та додавання індивідів до класів. Другий актор – користувач розробленим програмним додатком, наприклад викладач. Викладачу доступні наступні функції:

– пошук методичних ресурсів;
– перегляд методичних ресурсів.

4.1 Створення онтології

Створення онтології заключається в побудові ієрархії класів, встановленні зв’язків та наповнені класів екземплярами.

Класи інтерпретуються як множини, елементами яких є екземпляри. Вони описуються, використовуючи формальні конструкції, які декларують вимоги для членства в класі. Класи можуть бути організовані в ієрархію відносин виду «підклас – суперклас» (таксономія). Підкласи є підмножинами свого суперкласу.

Задача правильного моделювання полягає в пошуку практично обґрунтованого та оптимального співвідношення між рівнем деталізації моделі (і, як наслідок, достовірності результатів моделювання) і необхідних для цього ресурсів [25].

Враховуючи те, що обрана предметна область, «Реєстр методичних ресурсів кафедри АПЕПС», були визначені наступні класи і їх підкласи. В якості основного класу виступає клас «Спеціальність» що має підкласи у вигляді різних документів. Так само маємо такі класи як «Викладачі» і «Дисципліни».

Для опису предметної області та структуризації методичних ресурсів використано наступні класи: спеціальність, викладачі, предмет, нормативні документи, галузеві стандарти, приймальна комісія, навчальні матеріали та програми, кожен з яких містить дані з різним смисловим значенням. Клас «Спеціальність» містить перелік спеціальностей, що є на кафедрі. Цей клас використано для зручності використання пошукової системи. Клас «Викладачі» містить список викладачів кафедри, кожен з яких є автором певного ресурсу, наприклад, переліку питань до іспиту. Клас «Предмет» містить перелік дисциплін базової підготовки. Клас «Нормативні документи» складається з двох підкласів «Навчальні плани» та «Навчальні програми». У класі містяться ресурси, які відносяться до опису роботи навчального процесу на кафедрі. Клас «Галузеві стандарти» ділиться на два підкласи «ОКХ» та «ОПП». Клас «Приймальна комісія» містить два підкласи «Правила роботи приймальної комісії» та «Аналіз результатів вступної кампанії», які описують роботу приймальної комісії та результати вступної кампанії відповідно. Для зберігання матеріалів необхідних для студента створено клас «Навчальні матеріали», який у свою чергу складається з підкласів: «Екзаменаційні білети», «Самостійні роботи», «Лекції», «Практичні завдання», «Методички», «Комплексні контрольні роботи», «Дистанційне навчання», «Поточний контроль», «Семестровий контроль». Клас «Програми» містить чотири підкласи: «Анотації», «Навчальні програми», «Робочі програми», «Рейтингова система», які містять характеристику про дисципліну та технологію її викладання, а також систему формування рейтингу.

Таким чином створюємо всі необхідні класи. Ієрархію класів показано на рисунку 4.3.

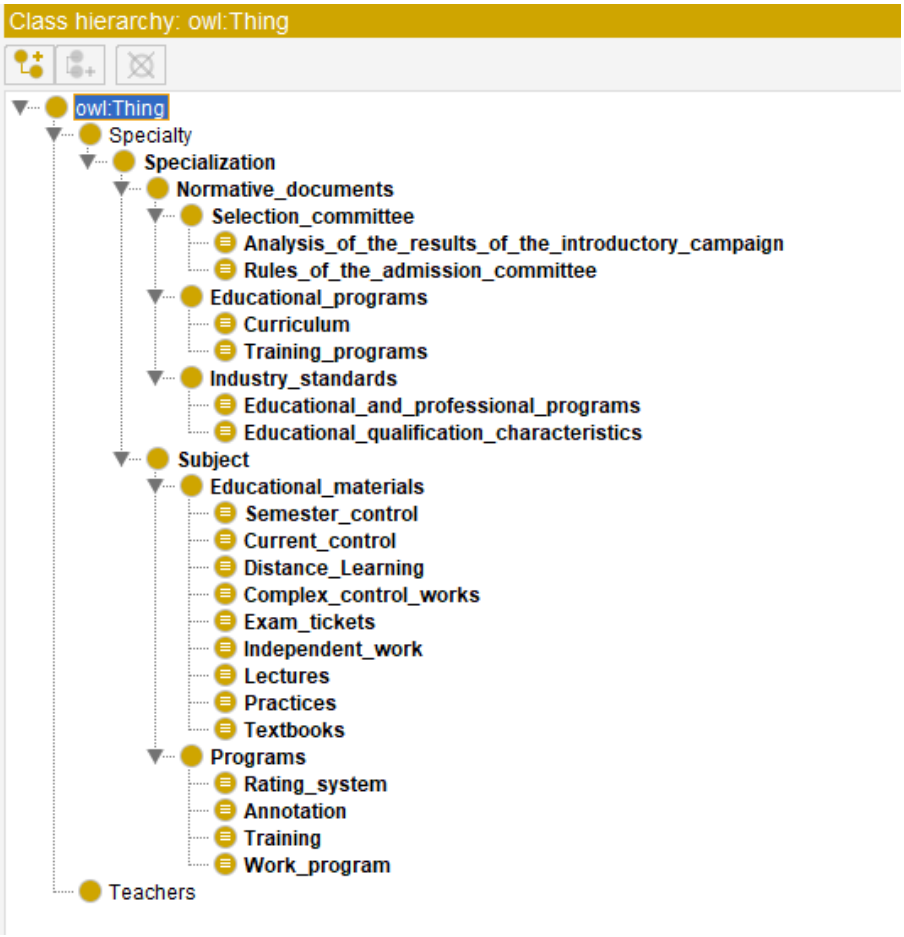


Рисунок 4.3 – Ієрархія класів

Отриманий результат також можна побачити у вигляді графа на закладці «OntoGraf» (рисунок 4.4):

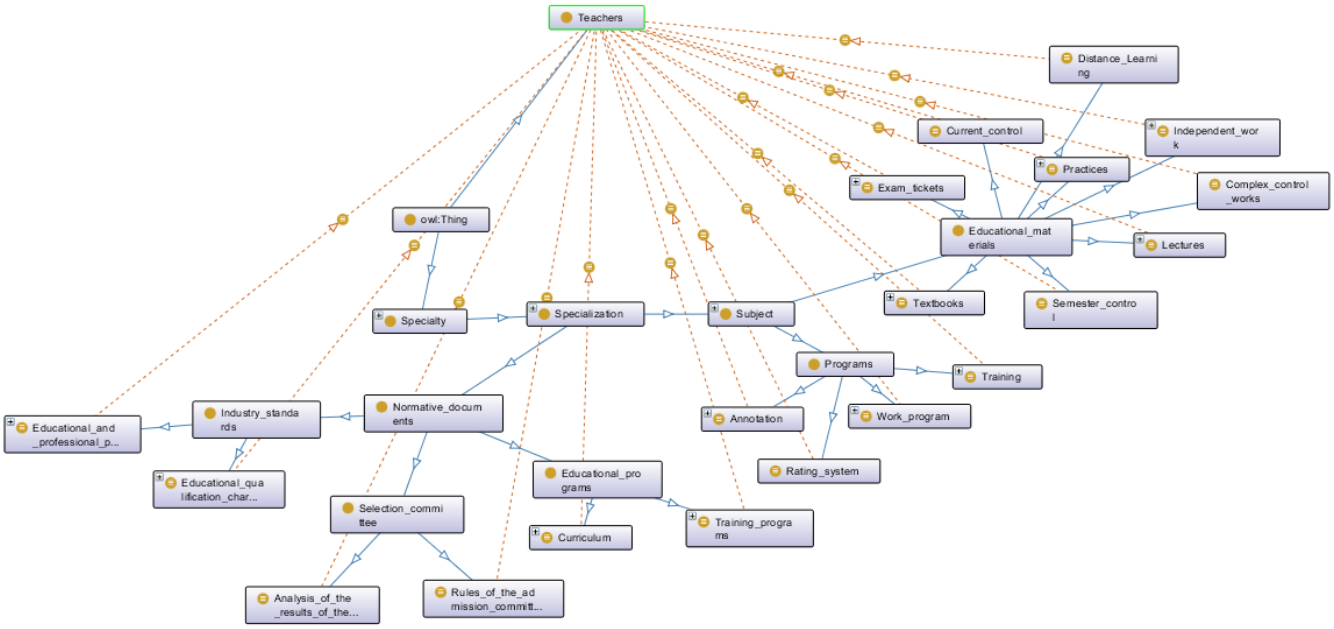


Рисунок 4.4 – Дерево класів

Щоб описати об’єкти та робити логічні висновки на підставі нашої моделі, потрібно, щоб були присутні елементи як мінімум трьох інших видів – різні види угруповання об’єктів, опису їх властивостей і зв’язків.

Модель буде обов’язково містити сутності таких типів:

– визначення класів (груп) об’єктів;

– визначення статичних атрибутів об’єктів;

– визначення зв’язків між об'єктами;

– індивідуальні (конкретні) об’єкти.

Після створення класів необхідно потурбуватися про створення властивостей. Розглянемо більш детально це поняття.

У Protege існує два основних типи властивостей:

– властивості об’єкта (Object Properties);

– властивості типів даних (Data Properties).

Властивостями об’єкта позначають відносини між двома екземплярами. Остаточний список властивостей об’єктів представлений на рисунку 4.5.

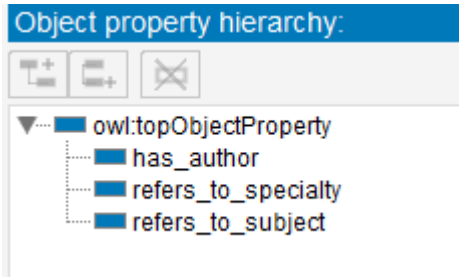


Рисунок 4.5 – Список властивостей об’єктів.

Властивості типів даних (атрибути класу) – описують зв'язки між індивідом і значення даних.

Процес створення аналогічний створенню класів і властивостей об’єкта. Остаточний список властивостей типів даних наведено на рисунку 4.6.

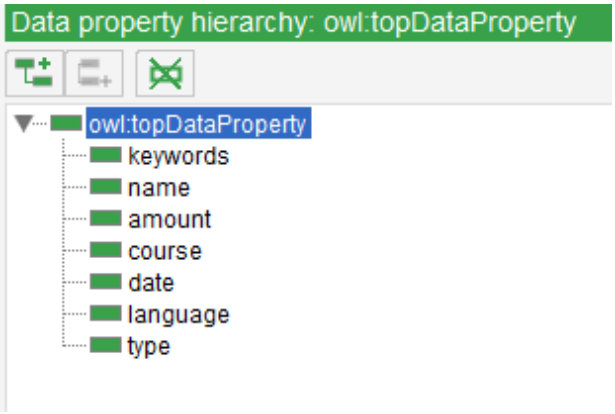


Рисунок 4.6 – Властивості типів даних

Розглянемо більш детально метадані:

- найменування – назва, надана інформаційному ресурсу його створювачем, власником або укладачем метаописання, при відсутності назви.
- автор – створювач ресурсу.
- ключові слова – вказують ключові слова, що описують тематику інформаційному ресурсу.
- мова – мова, застосована для подання текстових даних в інформаційному ресурсі.
- дата створення – рік створення інформаційного ресурсу, чотиризначне число.
- тип – тип файлу.

– курс – належність до курсу (1-4).
– обсяг – характеристика обсягу EIP. Приводять у мегабайтах. Приклад: 14 Мбайт, 279 Мбайт, 1048 Мбайт.

Заключним в створенні онтології є заповнення класів екземплярами.

На рисунку 4.7 показано наповнення класу «Exam_tickets» екземплярами.

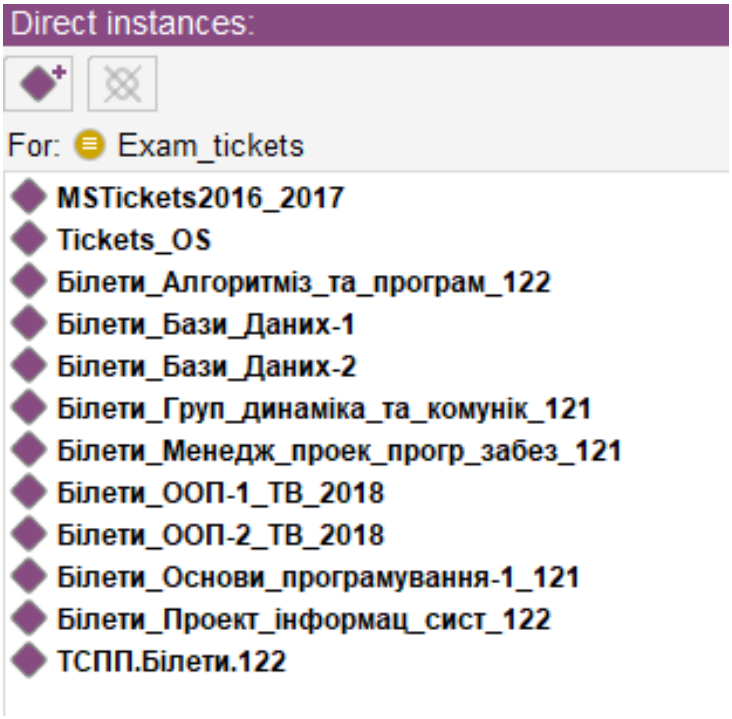


Рисунок 4.7 – Створення екземпляру об’єкту

Після створення індивідуалів потрібно було задати властивості об’єктів між екземплярами класів. Після цього потрібно було заповнити атрибути для всіх класів. При заповненні атрибутів для класу потрібно вказувати тип даних.

На рисунку 4.8 наведено приклад встановлення атрибутів для екземпляру «Моделювання систем у середовищі MATLAB» класу «Textbooks».

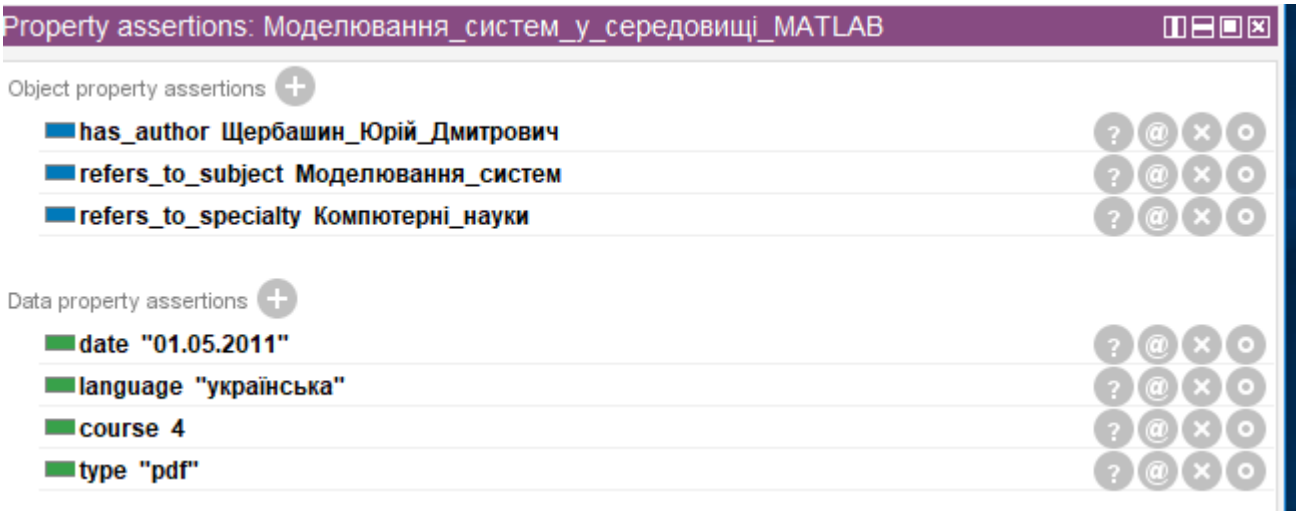


Рисунок 4.8 – Приклад атрибутів для екземпляру

Для цієї онтології для кожного атрибута використовується тип даних «string» так як він є універсальним типом даних, який дозволяє використовувати як цифри так і символи.

4.2 Написання запитів до розробленої онтології

Для написання клієнтського застосунку будуть використані шаблони для SPARQL-запитів. Перш ніж приступити до розробки застосунку необхідно написати запити та протестувати їх а середовищі Protege. У системі використовується 10 типів запитів. Розглянемо більш детально деякі з них.

Першим типом запитів є запит, який дозволяє вивести всі методичні ресурси по спеціальності та курсу, які належать до Educational materials.

На рисунку 4.9 наведено приклад запиту вже з конкретним курсом та спеціальністю. У даному випадку вибираються матеріали, що відносяться до спеціальності «Інженерія програмного забезпечення» та другого курсу.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#>
SELECT ?o
WHERE {
  ?o rdf:type ?type.
  ?type rdfs:subClassOf* metodicalres:Educational_materials.
?o metodicalres:refers_to_specialty
metodicalres:Інженерія_програмного_забезпечення.
?o  metodicalres:course ?course.
filter ( ?course = 2)
}
```

Рисунок 4.9 – Приклад запиту для пошуку по спеціальності та курсу

Наступний тип запиту дозволяє отримати ресурси, що відносяться до певного викладача та мають певний тип документу (рисунок 4.10).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#>
SELECT ?o
WHERE {
  ?o rdf:type ?type.
  ?type rdfs:subClassOf* metodicalres:Educational_materials.
?o metodicalres:has_author  metodicalres:Кублій_Лариса_Іванівна.
?o metodicalres:language ?language.
FILTER regex(?language, "українська")
}
```

Рисунок 4.10 – Приклад запиту для пошуку по викладачу та типу документу

Третім типом запиту є запит за допомогою якого можна отримати ресурси, що задовольняють умовам вибраної дисципліни та курсу (рисунок 4.12).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#>
SELECT ?o
WHERE {
  ?o rdf:type ?type.
  ?type rdfs:subClassOf* metodicalres:Educational_materials.
?o metodicalres:refers_to_subject  metodicalres:Системний_аналіз.
?o  metodicalres:course ?course.
filter ( ?course = 2)
}
```

Рисунок 4.12 –Приклад запиту для отримання ресурсів по дисципліні та курсу

Четвертим типом є запит для отримання ОКХ за спеціальністю (рисунок 4.13).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#>
SELECT ?o
WHERE {
  ?o rdf:type ?type.
  ?type rdfs:subClassOf*
metodicalres:Educational_qualification_characteristics.
 ?o metodicalres:refers_to_specialty  metodicalres:Компютерні_науки.
}
```

Рисунок 4.13 – Приклад запиту для отримання ОКХ за спеціальністю

П’ятий тип запиту виводить навчальні плани за вказаним курсом, наприклад курс рівний 2 (рисунок 4.14).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#>
SELECT ?o
WHERE {
  ?o rdf:type ?type.
  ?type rdfs:subClassOf* metodicalres:Curriculum.
?o  metodicalres:course ?course.
filter ( ?course = 2)
}
```

Рисунок 4.14 – Приклад запиту для отримання навчальних планів за курсом

Як видно з рисунків, у запитах вказані конкретні назви дисциплін, конкретний курс, спеціальність та викладач. Клієнтський додаток розроблено таким чином, що в запит замість конкретних назв можна підставити будь-яку іншу за бажанням і потребою користувача.

4.3 Створення клієнтського застосунку

Клієнтський застосунок написано за допомогою платформи .NET Framework мовою C# та з використанням бібліотеки dotNetRDF.

Мова C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Бібліотека DotNetRDF – повна бібліотека для аналізу, управління, запитів і запису RDF.

За допомогою вище згаданої бібліотеки відбувається робота з owl-файлом. Для цього необхідно вказати назву файлу та шлях до нього. Після визначення файлу

створено методи, що дозволяють обробляти отримані користувачем дані та створюють SPARQL-запити.

Основними методами програми є TrainingProgramCurriculum, OkhOpp, FindBySubject, FindByTwo, OpenFile. Перші чотири методи застосовуються для виконання запитів. Кожен метод використовується для конкретного типу запитів. Метод TrainingProgramCurriculum виводить навчальні плани та навчальні програми по курсу. Метод OkhOpp знаходить всі ОКХ та ОПП за спеціальністю. Метод FindBySubject знаходить навчальні програми, робочі програми та анотації по дисципліні. Метод FindByTwo знаходить всі інші методичні ресурси, метод виконує пошук по двом критеріям, саме тому має таку назву. Метод OpenFile дозволяє відкрити файл для його перегляду.

В якості графічного інтерфейсу створено форму Windows Form з усіма необхідними полями та кнопками. Після чого було написано обробники подій для кнопок, в яких викликаються вищезгадані методи. Це надає змогу виконати запит та отримати результат у зручному для користувача вигляді.

4.4 Висновки до розділу

У розділі обґрунтована побудова класів, які описують метадані реєстру методичного забезпечення інформаційними ресурсами кафедри. Дано перелік атрибутів до класів: автор, дата створення документу, курс, мова документу та інші.

Також у розділі описано функції програмного продукту.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання.

5.1 Системні вимоги та інсталяція

Для встановлення розробленої програмної системи персональний комп'ютер повинен мати процесор Intel ® Core ™ 2 / 2 Quad / Pentium ® / Celeron ® / Xeon™ чи AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™з тактовою частотою не нижче 2 GHz, на комп'ютері повинна бути встановлена операційна система Windows 10, Windows 8, Windows 7. Крім того, на персональному комп'ютері повинні бути встановлені Microsoft Office 2007 або більш нові версії, Microsoft .NET Framework 4, а також Protégé 5.5 або інша версія.

Для запуску та експлуатації розробленої системи інсталяція не потрібна, достатньо запустити виконуваний файл MetodicalRes.exe, після цього система запуститься і з'явиться вікно програми.

5.2 Сценарій роботи користувача з системою

Користувач взаємодіє з графічним інтерфейсом, що доступний як десктопний додаток. Оскільки авторизація при запуску програми не відбувається, то користувач одразу бачить робочу область програми в якості першого вікна. Авторизація не відбувається, оскільки система не працює з даними, що мають обмежений доступ. Саме тому зберігання інформації про користувача недоцільно. На рисунку 5.1 зображено головне вікно програми.

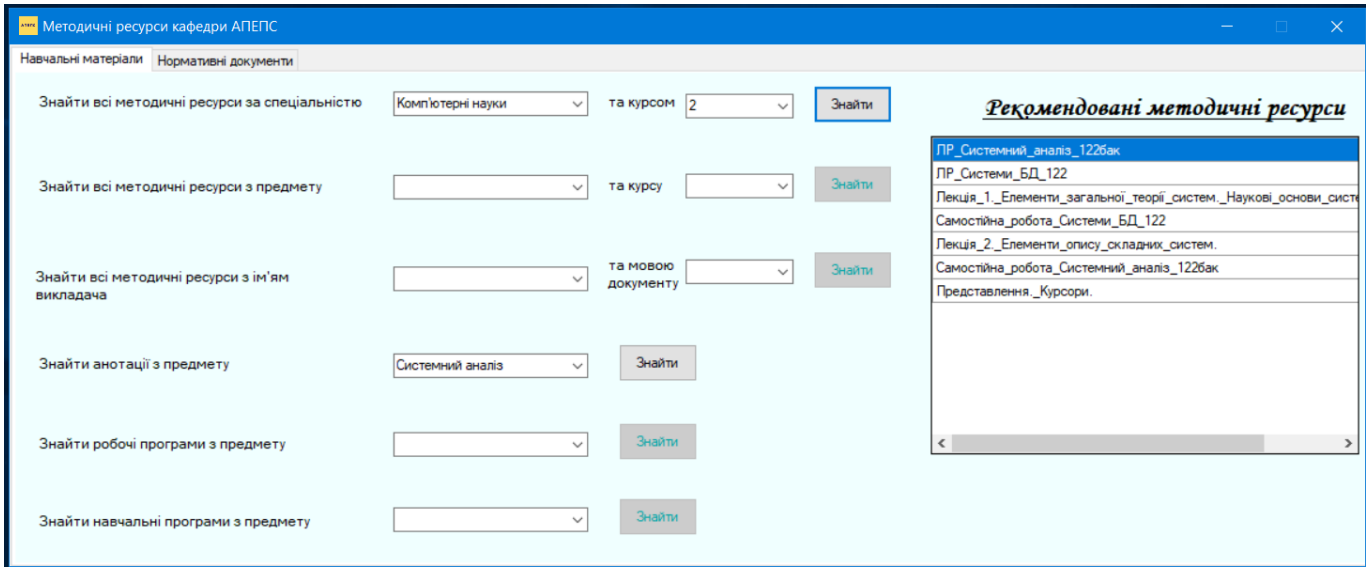


Рисунок 5.1 – Головне вікно програми

Додаток містить дві вкладки:

– навчальні матеріали;

– нормативні документи.

Перша вкладка дає змогу переглядати методичні ресурси, що стосуються дисципліни і буде більш доцільною для використання студентом. На вкладці розміщені пропоновані запити, що дають змогу отримати бажаний ресурс. Користувачу необхідно лише заповнити пробіли необхідною інформацією. Для полегшення взаємодії користувача з системою фільтри розроблені у вигляді випадаючого списку, що дозволяє уникнути, наприклад, лексичних помилок. Після заповнення полів розблокується кнопка «Знайти». Для уникнення виконання порожніх запитів кнопка «Знайти» заблокована до тих пір поки користувач не заповнить всі пробіли в запиті. Натиснувши кнопку, у вікні зправа з’явиться запропонований список методичних ресурсів. Приклад результату пошуку наведено на рисунку 5.2.

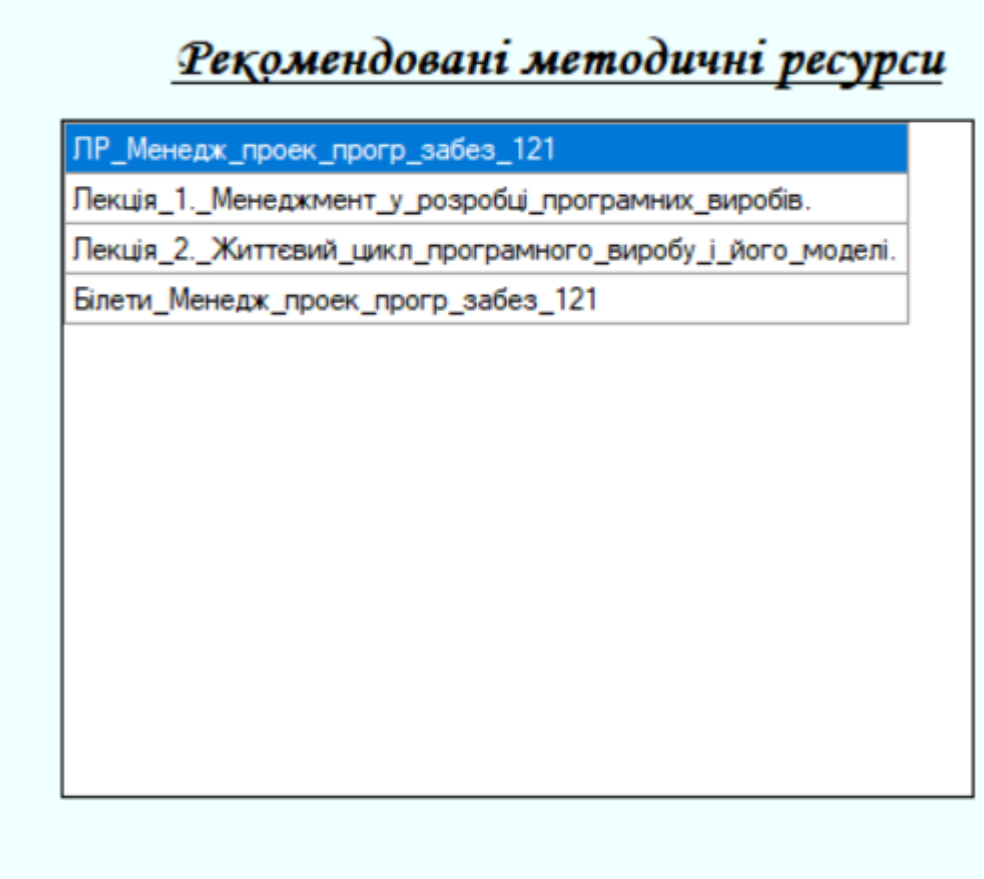


Рисунок 5.2 – Результат пошуку

Програма надає можливість не лише переглянути список матеріалів, а й відкрити їх у програмі Microsoft Office. Для цього необхідно двічі натиснути на назві ресурсу. Результат відкриття документу наведено на рисунку 5.3.

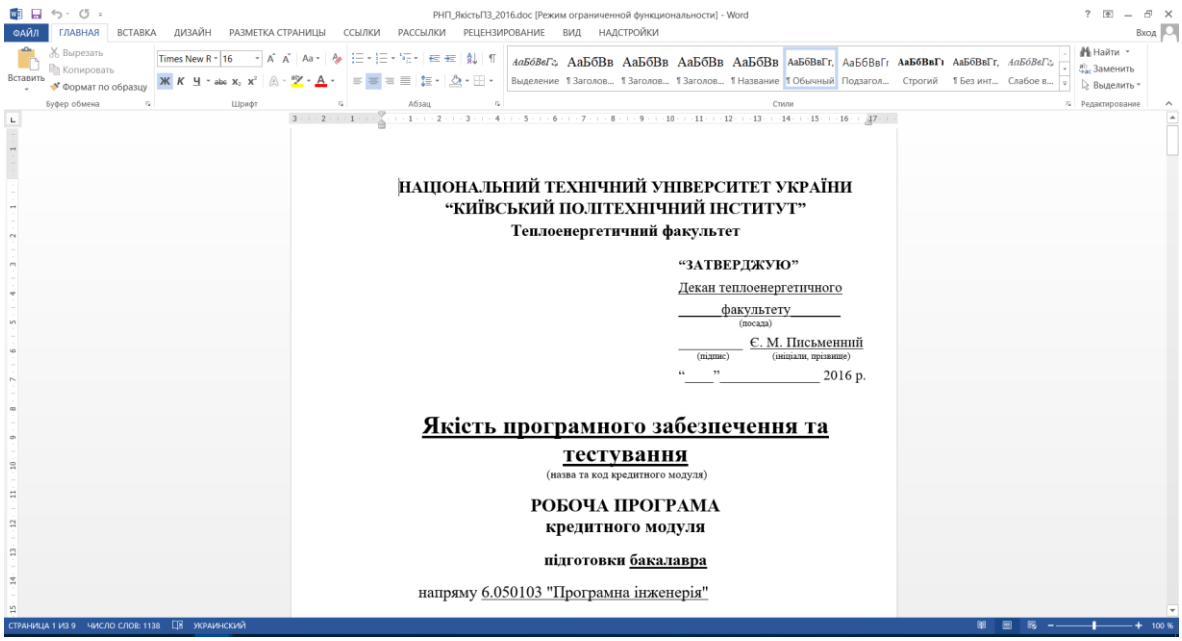


Рисунок 5.3 – Результат відкриття документу

Оскільки дана система, з часом, буде наповнюватися методичними ресурсами, то може виникнути ситуація, коли відкрити документ, на даний момент, не вдасться. У такому випадку користувачу буде показано вікно з повідомленням (рисунок 5.4).

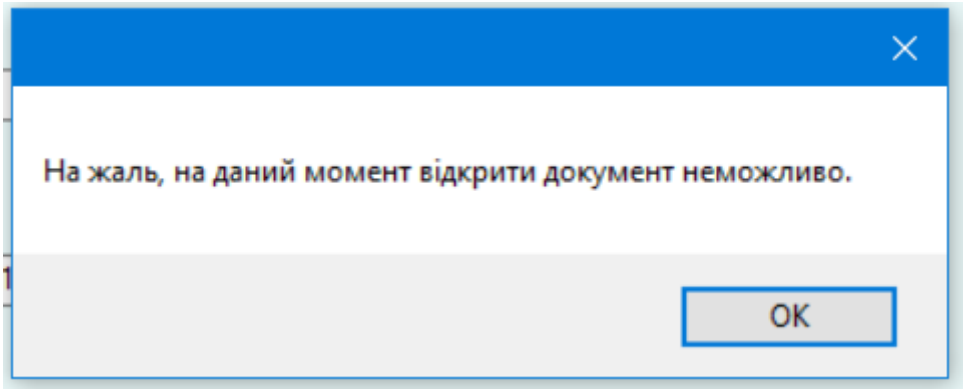


Рисунок 5.4 – Повідомлення про відсутність ресурсу

Вкладка «Нормативні документи» дозволяє переглянути нормативні документи, такі як ОКХ та ОПП, і буде більш доцільна для працівників кафедри. Робота з вкладкою аналогічна до роботи з попередньою вкладкою. Інтерфейс другої вкладки наведено на рисунку 5.5.

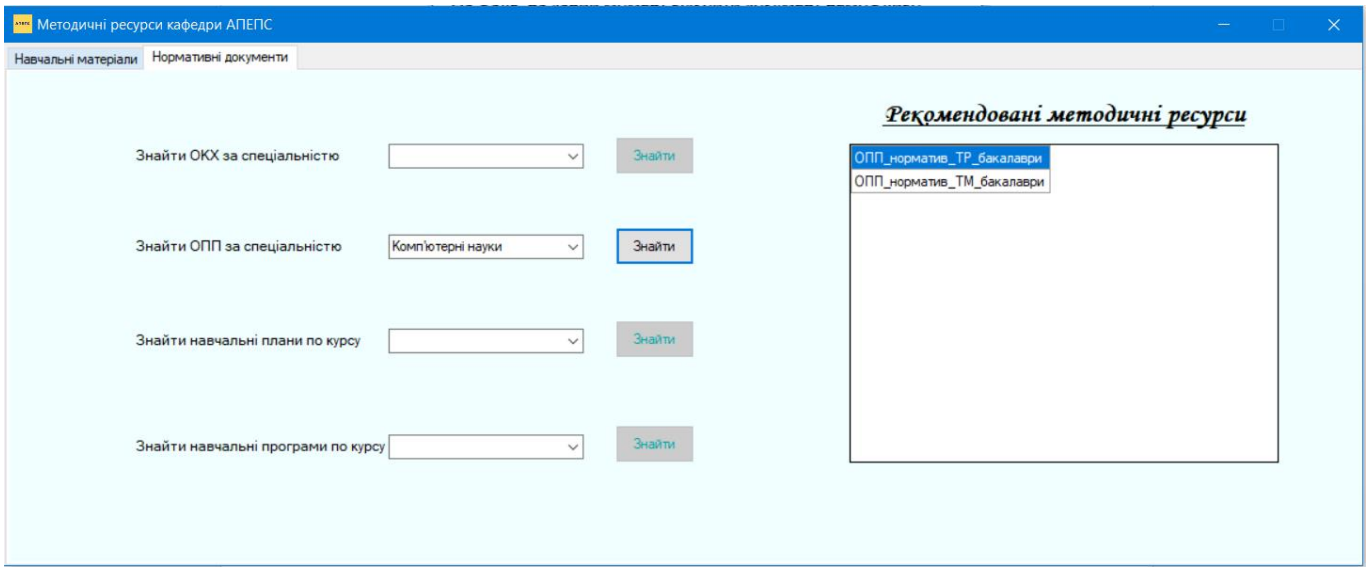


Рисунок 5.4 – Інтерфейс вкладки «Нормативні документи»

Крім того, користувач може переглянути онтологію. Для цього необхідно відкрити файл metodicalRes.owl в програмі Protégé.

Користувач може переглянути список класів, об’єктів, властивостей та зв’язків. Користувач може перейти на пункт меню «Зв’язки даних» та побачити всі створені зв’язки та між якими об’єктами вони працюють. (рисунок 5.5).

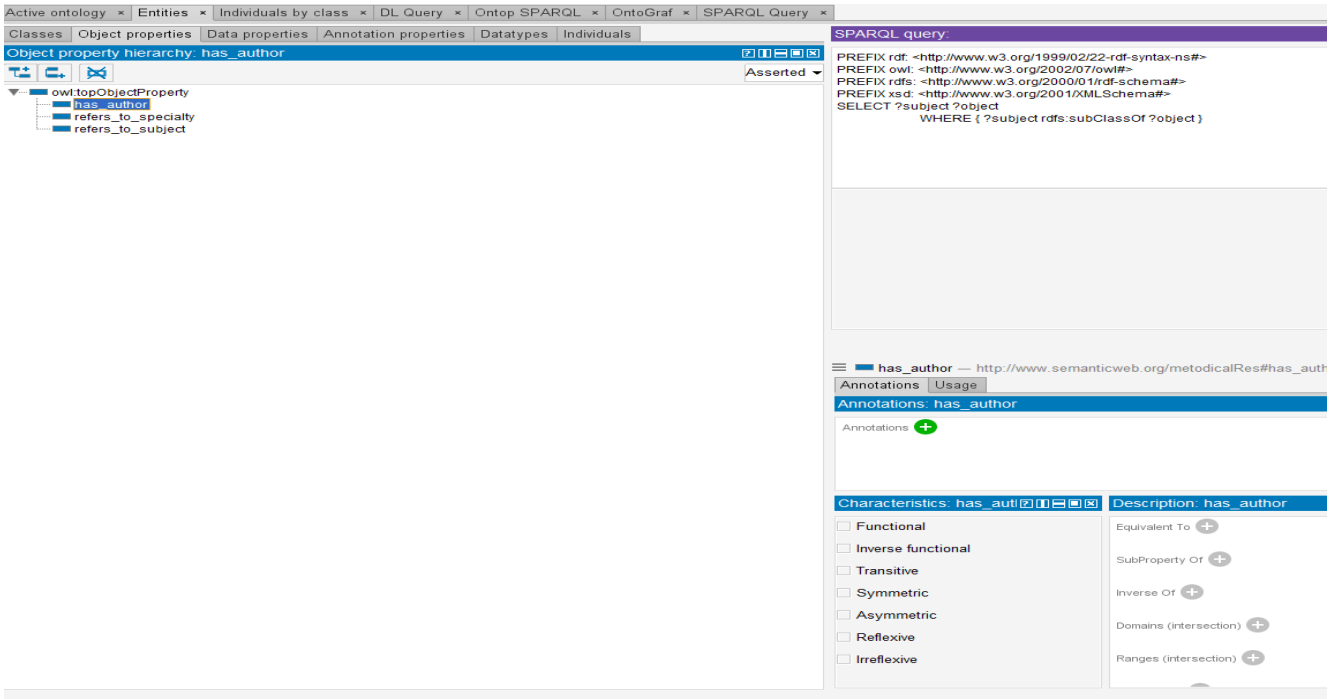


Рисунок 5.5 – Вікно «Зв’язки даних»

Також можна перейти на вкладку «Individuals by class» і переглянути всі об’єкти які були створенні в певному класі. На рисунку 5.6 наведено приклад індивідів, що містяться в класі Exem_tickets (екзаменаційні білети). Клас містить 12 індивідів.

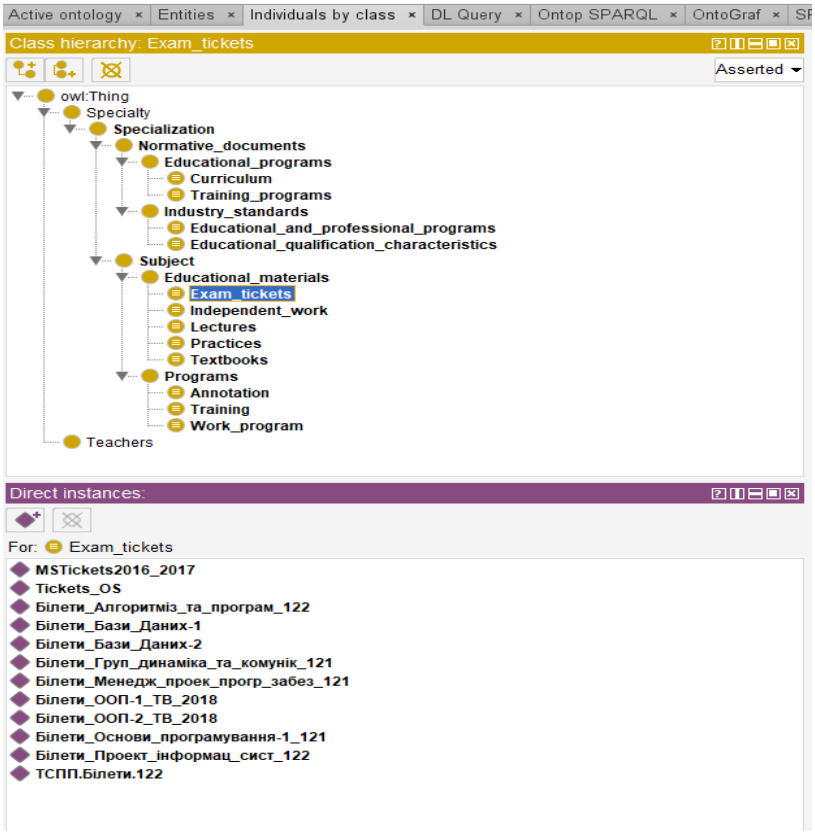


Рисунок 5.6 – Вікно «Individuals by class»

Крім того, користувач має можливість додати нові об’єкти до кожного з класів, тим самим збільшити наповнення онтології. Однією з проблем створення нового індивіда є необхідність вказувати зв’язки, що є доволі громіздким процесом та займає достатньо багато часу.

За допомогою клієнтського застосунку відбувається доступ до ресурсів, що містяться в репозиторії.

Спираючись на сукупність всіх раніше згаданих можливостей системи, можна переглядати наповненість дисципліни методичним забезпеченням та редагувати її.

5.3 Висновки до розділу

У розділі описано принцип роботи з системою. Важливо відмітити, що система складається з двох частин: розробленої онтології та клієнтського застосунку. Користувач має можливість взаємодіяти з обома частинами. У розділі перераховані та описані функції, які може виконувати система.

ВИСНОВКИ

Отже, під час виконання дипломної роботи було реалізовано наступні задачі:

1. Проведено дослідження предметної області: було вивчено роботу кафедри, її основні структурні об’єкти, взаємозв’язки між ними.

2. Розроблено онтологію реєстру методичних ресурсів кафедри, яка включає в себе 21 клас, 3 найчастіше використовуваних зв’язки між класами та 4 властивості. Дана онтологія значно спрощує роботу персоналу для пошуку та обробки даних кафедри.

3. Написано SPARQL-запити для пошуку по онтології.

4. Також були проаналізовані існуючі програмні засоби візуальної побудови опису онтологій та створений програмний продукт, що надає змогу виконувати пошук по онтології без написання складних SPARQL-запитів користувачем. Визначені поняття онтології та SPARQL-запиту.

5. Розроблена система надає можливість переглядати документи за допомогою програми Microsoft Office.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ковтанюк Ю.С., Макарченко П.М., Дубровіна Л.А. Описування та організація зберігання електронних інформаційних ресурсів органів державної влади, органів місцевого самоврядування, підприємств, установ і організацій: методичні рекомендації / Укрдержархів України, УНДІАСД : Київ, 2010. – 30 с.

2. Тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського: 5.4. Навчально-методичне забезпечення навчальних дисциплін.

3. Guarino N. What Is an Ontology? / N. Guarino, D. Oberle, S. Staab. // International Handbooks on Information Systems. – 2009.

4. Benjamins V.R. An Intelligent Brokering Service for Knowledge-Component Reuse on the World – Wide Web / Benjamins V.R. // Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, June 2009, Чикаго, США , матеріали. — Ч.: W360, 2009 С. 125-129.

5. Basu D. Smart Doorplate / Basu D. // Journal of Ontologies. – 2003. – №25. – С. 201-216.

6. Структура онтологии [Електронний ресурс] — Режим доступу: https://bitly.su/G0Cn5qk.

7. Основные понятия онтологии: Научная электронная библиотека [Електронний ресурс] — Режим доступу: https://monographies.ru/ru/book/section?id=11245.

8. RDF Current Status - W3C [Електронний ресурс] // W3C. – 2015. – Режим доступу: https://www.w3.org/standards/techs/rdf#w3c_all.

9. S. Aliyu. A Category Theoretic Model of RDF Ontology / S. Aliyu, S.B. Junaidu, A. F. Donfack Kana. // International Journal of Web & Semantic Technology (IJWesT). – 2015. – №3.

10. RDF 1.1 Semantics [Електронний ресурс] // W3C. – 2014. – Режим доступу: https://www.w3.org/TR/rdf11-mt/.

11. Онтология в компьютерных системах [Електронний ресурс] — Режим доступу: https://rsdn.org/article/philosophy/what-is-onto.xml.

12. Horrocks I. From SHIQ and RDF to OWL: The Making of a Web Ontology Language / I. Horrocks, P. Patel-Schneider, F. van Harmelen.

13. SPARQL 1.1 Query Language, S. Harris, A. Seaborne, Editors, W3C Recommendation [Електронний ресурс] – 21 March 2013. — Режим доступу: http://www.w3.org/TR/sparql11-query.

14. Kollia I. SPARQL Query Answering over OWL Ontologies / I. Kollia, B. Glimm, I. Horrocks. // Oxford University Computing Laboratory.

15. Y. Sure. OntoEdit: Collaborative ontology development for the Semantic Web. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke // In Proc. of the Inter. Semantic Web Conference (ISWC 2002), Sardinia, Italia, June 2002.

16. Bechhofer S.,Horrocks I., Goble C., Stevens R. OilEd: A Reason-able Ontology Editor for the Semantic Web // Joint German/Austrian conf. on Artificial Intelligence (KI'01). Lecture Notes in Artificial Intelligence LNAI 2174, Springer-Verlag, Berlin, pages.396-408, 2001.

17. Domingue J. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web // Proc. of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff, Canada, 1998.

18. Semantic Infrastructure OntoStudio [Електронний ресурс] — Режим доступу: http://www.semafora-systems.com/.

19. Система Apollo [Електронний ресурс] — Режим доступу: http://apollo.open.ac.uk/index.html

20. Еволюція Protégé [Електронний ресурс] — Режим доступу: https://studfiles.net/preview/5064224/page:61/.

21. Сайт програми Protégé Microsoft [Електронний ресурс] — Режим доступу: https://protege.stanford.edu/products.php.

22. Муромцев Д.И. Онтологический инжиниринг знаний в системе Protégé: методическое пособие – СПб:СПб ГУ ИТМО: 2007. – 62 с.

23. Visual Studio 2015 – новая версия среды разработки от компании Microsoft [Електронний ресурс] — Режим доступу: https://info-comp.ru/novosti/477-visual-studio-2015.html.

24. Общие сведения о платформе .NET Framework [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview.

25. Горшков С. «Введение в онтологическое моделирование», ООО «ТриниДата», 2014-2016

ДОДАТОК 1

Онтологічне представлення реєстру методичних ресурсів кафедри

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52 19Б 81-1	Бублик_А.С_ТМ52.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52 19Б 12-1	Program.cs	Модулі клієнтської частини програмного продукту
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52 19Б 12-2	Form1.cs	Модуль інтерфейсу клієнтської частини
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52 19Б 13-2	Опис.docx	Опис модуля інтерфейсу клієнтської частини програми

ДОДАТОК 2

Онтологічне представлення реєстру методичних ресурсів кафедри

Текст програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б 12-1

Аркушів 10

Київ – 2019

```
//підключення бібліотек
using System;
using System.Windows.Forms;
using VDS.RDF;
using VDS.RDF.Query;

namespace MetodicalRes
{
   public partial class Form1 : Form
   {
      //конструктор
      public Form1()
      {
         InitializeComponent();
      }
      //поля
      string newCombText;
      string query;
      //функція для отримання назв об'єктів з онтології без префіксів
      private string SplitFunction(string a)
      {
         string[] mas = a.Split('#');
         return mas[1].ToString();
      }
      //методи для виконання запитів
      //9.      Вивести навчальні плани по курсу, які належать до Educational program.
      //10.     Вивести навчальні програми по курсу, які належать до Educational
program
      private void TrainingProgramCurriculum(string clas, string course)
      {
         string result=null;
         IGraph g = new Graph();
         g.LoadFromFile("D:\\study\\диплом\\My diploma\\metodicalRes.owl");
         try
         {
            Object results = g.ExecuteQuery("PREFIX rdf:<http://www.w3.org/1999/02/22-
rdf-syntax-ns#>  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX
metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o
rdf:type ?type.  ?type rdfs:subClassOf* metodicalres:"+ clas +". ?o  metodicalres:course
?course. filter(?course = "+ course+")}");
            if (results is SparqlResultSet)
            {
               //SELECT/ASK queries give a SparqlResultSet
               SparqlResultSet rset = (SparqlResultSet)results;
```

```
            foreach (SparqlResult r in rset)
            {
               result= SplitFunction(System.Web.HttpUtility.UrlDecode(r.ToString()));

                  dataGridView1.Rows.Add(result);
            }
            if (result == null)
            {
               MessageBox.Show("На даний момент матеріали відсутні.");
            }
         }
         else
         {
            //If you don't get a SparqlResutlSet or IGraph something went wrong
            //but didn't throw an exception so you should handle it here
            MessageBox.Show("ERROR");
         }
      }
      catch (RdfQueryException queryEx)
      {
         //There was an error executing the query so handle it here
         MessageBox.Show(queryEx.Message);
      }
   }
   //методи для виконання запитів
   //4.      Вивести ОКХ по спеціальності Компютерні_науки
   //5.      Вивести ОПП по спеціальності Компютерні_науки
   private void OkhOpp(string clas, string spec)
   {
      string result = null;
      IGraph g = new Graph();
      g.LoadFromFile("D:\\study\\диплом\\My diploma\\metodicalRes.owl");
      try
      {
         Object results = g.ExecuteQuery("PREFIX rdf:<http://www.w3.org/1999/02/22-
rdf-syntax-ns#>  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX
metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o
rdf:type ?type.?type rdfs:subClassOf* metodicalres:"+clas+". ?o
metodicalres:refers_to_specialty metodicalres:"+spec+".}");
         if (results is SparqlResultSet)
         {
            //SELECT/ASK queries give a SparqlResultSet
            SparqlResultSet rset = (SparqlResultSet)results;
            foreach (SparqlResult r in rset)
```

```
            {
                result = SplitFunction(System.Web.HttpUtility.UrlDecode(r.ToString()));

                dataGridView1.Rows.Add(result);
            }
            if (result == null)
            {
                MessageBox.Show("На даний момент матеріали відсутні.");
            }
        }
        else
        {
            //If you don't get a SparqlResutlSet or IGraph something went wrong
            //but didn't throw an exception so you should handle it here
            MessageBox.Show("ERROR");
        }
    }
    catch (RdfQueryException queryEx)
    {
        //There was an error executing the query so handle it here
        MessageBox.Show(queryEx.Message);
    }
}
//методи для виконання запитів
//6.    Вивести навчальні програми по предмету , які належать до Programs - Training
//7.    Вивести робочі програми по предмету, які належать до Programs - Work_program
//8.    Вивести анотації по предмету, які належать до Programs - Annotation
private void FindBySubject(string clas, string subject)
{
    string result = null;
    IGraph g = new Graph();
    g.LoadFromFile("D:\\study\\диплом\\My diploma\\metodicalRes.owl");
    try
    {
        Object results = g.ExecuteQuery("PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o rdf:type ?type.?type rdfs:subClassOf* metodicalres:"+clas+". ?o metodicalres:refers_to_subject metodicalres:"+subject+".}");
        if (results is SparqlResultSet)
        {
            //SELECT/ASK queries give a SparqlResultSet
```

```
            SparqlResultSet rset = (SparqlResultSet)results;
            foreach (SparqlResult r in rset)
            {
               result = SplitFunction(System.Web.HttpUtility.UrlDecode(r.ToString()));

               dataGridView2.Rows.Add(result);
            }
            if (result == null)
            {
               MessageBox.Show("На даний момент матеріали відсутні.");
            }
         }
         else
         {
            //If you don't get a SparqlResutlSet or IGraph something went wrong
            //but didn't throw an exception so you should handle it here
            MessageBox.Show("ERROR");
         }
      }
      catch (RdfQueryException queryEx)
      {
         //There was an error executing the query so handle it here
         MessageBox.Show(queryEx.Message);
      }
   }
   //методи для виконання запитів
   //1.    Вивести всі методичні ресурси по спеціальності та курсу, які належать
до Educational materials
   //2.    Вивести всі методичні ресурси з викладачем та мовою документу, які
належать до Educational materials
   //3.    Вивести всі методичні ресурси по предмету  та курсу, які належать до
Educational materials
   private void FindByTwo(string s)
   {
      string result = null;
      IGraph g = new Graph();
      g.LoadFromFile("D:\\study\\диплом\\My diploma\\metodicalRes.owl");
      try
      {
         Object results = g.ExecuteQuery(s);
         if (results is SparqlResultSet)
         {
            //SELECT/ASK queries give a SparqlResultSet
            SparqlResultSet rset = (SparqlResultSet)results;
```

```
            foreach (SparqlResult r in rset)
            {
                result = SplitFunction(System.Web.HttpUtility.UrlDecode(r.ToString()));

                dataGridView2.Rows.Add(result);
            }
            if (result == null)
            {
                MessageBox.Show("На даний момент матеріали відсутні.");
            }
        }
        else
        {
            //If you don't get a SparqlResutlSet or IGraph something went wrong
            //but didn't throw an exception so you should handle it here
            MessageBox.Show("ERROR");
        }
    }
    catch (RdfQueryException queryEx)
    {
        //There was an error executing the query so handle it here
        MessageBox.Show(queryEx.Message);
    }
}
//заміна назви спецільностей
private string  Subject(ComboBox comb)
{
    if (comb.Text== "Об’єктно-орієнтоване програмування-1")
    {
        newCombText = "Обєктно-орієнтоване_програмування-1";
    }
    else if (comb.Text== "Об’єктно-орієнтоване програмування-2")
    {
        newCombText = "Обєктно-орієнтоване_програмування-2";
    }
    else if (comb.Text== "Комп'ютерні мережі")
    {
        newCombText = "Компютерні_мережі";
    }
    else
    {
        newCombText = comb.Text.Replace(' ', '_');
    }
    return newCombText;
```

```
}
//пошук навчальних програм
private void button10_Click(object sender, EventArgs e)
{
   dataGridView1.Rows.Clear();
   TrainingProgramCurriculum("Training_programs",comboBox12.Text);
}
//завантаження форми
private void Form1_Load(object sender, EventArgs e)
{
   button1.Enabled = false;
   button2.Enabled = false;
   button3.Enabled = false;
   button4.Enabled = false;
   button5.Enabled = false;
   button6.Enabled = false;
   button7.Enabled = false;
   button8.Enabled = false;
   button9.Enabled = false;
   button10.Enabled = false;
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox12_SelectedIndexChanged(object sender, EventArgs e)
{
   button10.Enabled = true;
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox13_SelectedIndexChanged(object sender, EventArgs e)
{
   button8.Enabled = true;
}
//пошук навчальних планів
private void button8_Click(object sender, EventArgs e)
{
   dataGridView1.Rows.Clear();
   TrainingProgramCurriculum("Curriculum", comboBox13.Text);
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox10_SelectedIndexChanged(object sender, EventArgs e)
{
   button9.Enabled = true;
}
//пошук ОПП
private void button9_Click(object sender, EventArgs e)
```

```
{
    dataGridView1.Rows.Clear();
    OkhOpp("Educational_and_professional_programs", Speciality(comboBox10));
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox11_SelectedIndexChanged(object sender, EventArgs e)
{
    button7.Enabled = true;
}
//пошук ОКХ
private void button7_Click(object sender, EventArgs e)
{
    dataGridView1.Rows.Clear();
    OkhOpp("Educational_qualification_characteristics", Speciality(comboBox11));
}
//заміна спеціальності
private string Speciality(ComboBox comb)
{
    if (comb.Text == "Комп'ютерні науки")
    {
        newCombText = "Компютерні_науки";
    }
    else
    {
        newCombText = comb.Text.Replace(' ', '_');
    }
    return newCombText;
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox9_SelectedIndexChanged(object sender, EventArgs e)
{
    button6.Enabled = true;
}
//пошук навчальних програм по предмету
private void button6_Click(object sender, EventArgs e)
{
    dataGridView2.Rows.Clear();
    FindBySubject("Training", Subject(comboBox9));
}
//якщо текст в комбобокс заповнено, то кнопка розблокується
private void comboBox7_SelectedIndexChanged(object sender, EventArgs e)
{
    button5.Enabled = true;
}
```

```
        //пошук робочих програм по предмету
        private void button5_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();
            FindBySubject("Work_program", Subject(comboBox7));
        }
        //якщо текст в комбобокс заповнено, то кнопка розблокується
        private void comboBox8_SelectedIndexChanged(object sender, EventArgs e)
        {
            button4.Enabled = true;
        }
        //пошук анотацій
        private void button4_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();
            FindBySubject("Annotation", Subject(comboBox8));
        }
        //якщо текст в комбобокс заповнено, то кнопка розблокується
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox1.Text.Length>0)
            {
                button1.Enabled = true;
            }
        }
        //пошук ресурсів по спеціальності
        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();
            query = "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX
rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX
metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o
rdf:type ?type. ?type rdfs:subClassOf* metodicalres:Educational_materials. ?o
metodicalres:refers_to_specialty metodicalres:"+ Speciality(comboBox1) + ". ?o
metodicalres:course ?course. filter(?course = "+comboBox2.Text+")} ";
            FindByTwo(query);
        }
        //якщо текст в комбобокс заповнено, то кнопка розблокується
        private void comboBox5_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox6.Text.Length>0)
            {
                button3.Enabled = true;
            }
```

```
        }
        //пошук ресурсів по предмету
        private void button3_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();
            query = "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX
rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX
metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o
rdf:type ?type. ?type rdfs:subClassOf* metodicalres:Educational_materials. ?o
metodicalres:refers_to_subject  metodicalres:" + Subject(comboBox6) + ". ?o
metodicalres:course ?course. filter(?course = " + comboBox5.Text + ")} ";
            FindByTwo(query);
        }
        //якщо текст в комбобокс заповнено, то кнопка розблокується
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox4.Text.Length > 0)
            {
                button2.Enabled = true;
            }
        }
        //пошук ресурсів по викладачу
        private void button2_Click(object sender, EventArgs e)
        {
            dataGridView2.Rows.Clear();
            newCombText = comboBox4.Text.Replace(' ', '_');
            query = "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX
rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX
metodicalres:<http://www.semanticweb.org/metodicalRes#> SELECT ?o WHERE { ?o
rdf:type ?type. ?type rdfs:subClassOf* metodicalres:Educational_materials. ?o
metodicalres:has_author  metodicalres:"+newCombText+". ?o metodicalres:language
?language. FILTER regex(?language, '"+comboBox3.Text+"')}";
            FindByTwo(query);
        }
        //відкриття файлу
        private void OpenFile(string name)
        {
            try
            {
                string commandText = @"D:\\study\\диплом\\My diploma\\Методичні
ресурси\\" + name + ".doc";
                var proc = new System.Diagnostics.Process();
                proc.StartInfo.FileName = commandText;
                proc.StartInfo.UseShellExecute = true;
```

```
            proc.Start();
        }
        catch (Exception ex)
        {
            try
            {
                string commandText = @"D:\\study\\диплом\\My diploma\\Методичні ресурси\\" + name + ".docx";
                var proc = new System.Diagnostics.Process();
                proc.StartInfo.FileName = commandText;
                proc.StartInfo.UseShellExecute = true;
                proc.Start();
            }
            catch (Exception e)
            {
                MessageBox.Show("На жаль, на даний момент відкрити документ неможливо.");
            }
        }
    }
    //відкриття файлу
    private void dataGridView2_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        OpenFile(dataGridView2.CurrentCell.Value.ToString());
    }
    //відкриття файлу
    private void dataGridView1_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        OpenFile(dataGridView1.CurrentCell.Value.ToString());
    }
  }
}
```

ДОДАТОК 3

Онтологічне представлення реєстру методичних ресурсів кафедри

Опис програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б 13-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає інформацію про програмну систему онтологічного представлення реєстру методичних ресурсів кафедри.

Розроблене програмне забезпечення дозволяє отримувати концептуальну схему онтології, виконувати попередньо написані запити, отримувати та виводити результат.

Клієнтський застосунок було розроблено в середовищі Microsoft Visual Studio 2015 з використанням платформи .NET та мови C#.

ЗМІСТ

1. Загальні відомості 4
2. Функціональне призначення 5
3. Опис логічної структури 6
4. Використовувані технічні засоби 7
5. Вхідні і вихідні дані 8

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до теми дипломної роботи, програма має назву «Методичні ресурси кафедри АПЕПС».

Програма працює локально на комп’ютері користувача та не потребує доступу до мережі інтернет, що забезпечує практичність та швидкість.

Застосунок був написаний мовою C# з використанням бібліотеки dotNetRdf.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб покликаний вирішити задачу доступу до методичних ресурсів, використовуючи онтологічні джерела, щоб спростити роботу зі зв’язаними даними. Це було реалізовано за допомогою кількох функцій системи, а саме:

– генерація та виконання закладених в програму SPARQL-запитів. В програму закладені вже написані запити, тобто не потрібно розробляти нові запити;

– здійснення швидкого пошуку по онтології;

– можливість доступу до ресурсу. Знайдений ресурс можна відкрити та переглянути.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається з клієнтської та серверної частини.

Загальний принцип роботи додатку такий:

1) користувач змінює стан сторінки (наприклад, тисне на кнопку);
2) відбувається звернення до пов’язаного обробника подій;
3) метод, що викликається в обробнику виконує запит;
4) результати виконання методу виводяться на сторінку.

Продемонструвати роботу програми можна на прикладі виконання будь-якого запиту, наприклад для пошуку анотацій для дисципліни.

В першу чергу завантажується форма з усіма її компонентами: кнопки, текстові поля, таблиці.

Після натискання на кнопку, що розміщена біля тіла запиту, викликається метод FindBySubject, в який передається два параметри з самої форми – назва класу та назва дисципліни. Виконавшись, метод повертає список анотацій. Цей список виводиться до компоненту dataGridView. Після подвійного натискання на комірці dataGridView викликається метод OpenFile, який відкриває цей файл.

Після натискання кнопки для виконання іншого запиту dataGridView оновлюється та виводить інші дані.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для організації доступу до програмного продукту потрібно мати комп’ютер або ноутбук з операційною системою Windows.

Кінцевим користувачам для роботи з програмою потрібно, щоб на комп’ютері був встановлений Microsoft .NET Framework 4 або інша версія.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— owl-файл зі структурою онтології;

— шаблони запитів SPARQL.

Вихідними даними є:

— результати виконання запиту в табличній формі;

— посилання на методичний ресурс.

ДОДАТОК 4

Онтологічне представлення реєстру методичних ресурсів кафедри

Апробація

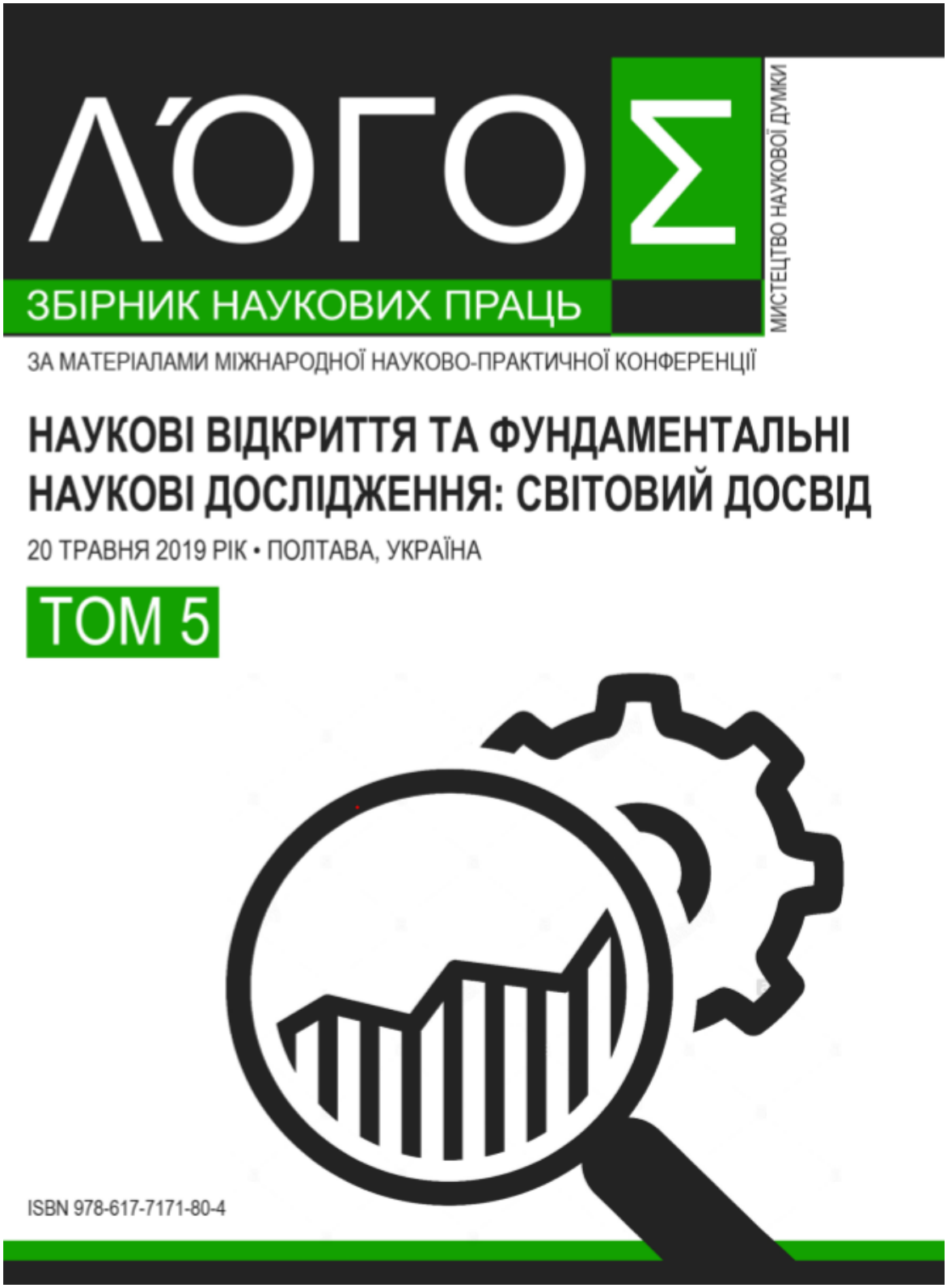
УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ52_19Б

Аркушів 7

Київ – 2019

-2-

Було подано тези щодо теми дипломної роботи до міжнародної науково-практичної конференції «Наукові відкриття та фундаментальні наукові дослідження: світовий досвід», що відбулася 20 травня 2019 року у м. Полтава, Україна. Тези надруковані в томі №5 на сторінці 60. Текст доповіді наведений нижче.



ОНТОЛОГІЧНЕ ПРЕДСТАВЛЕННЯ РЕЄСТРУ МЕТОДИЧНИХ РЕСУРСІВ КАФЕДРИ
Бублик А.С. **60**

ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНИХ ПРОДУКТІВ (ДОДАТКІВ) НА БАЗІ БЛОКЧЕЙН ТЕХНОЛОГІЇ
Марков Д.Е. **63**

ПРОГРАМНІ ЗАСОБИ НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ РОЗПІЗНАВАННЯ МОВЛЕННЯ WAV2LETTER++
Калиненко Д.О. **64**

СТАТИСТИЧНИЙ АНАЛІЗ ДИНАМІКИ АВІАЦІЙНОЇ ГАЛУЗІ В УКРАЇНІ ЗА 2018 РІК
Філіппова О.Ю., Глівінський Д.О. **65**

ФЛУОРЕСЦЕНТНІ ПІГМЕНТНІ ПРЕПАРАТИ ДЛЯ ДРУКУВАННЯ НА ТЕКСТИЛЬНИХ МАТЕРІАЛАХ
Мороз О.В. **68**

ФУНКЦІОНАЛЬНЕ ДІАГНОСТУВАННЯ ЕЛЕКТРОПРИВОДУ ШАХТНОЇ ПІДЙОМНОЇ МАШИНИ В РЕЖИМІ КЛАСТЕР-АНАЛІЗУ
Зимовець В.І. **72**

ЩОДО СТРАТЕГІЙ ОБСЛУГОВУВАННЯ ТЕХНІЧНОГО УСТАТКУВАННЯ МАГІСТРАЛЬНИХ ГАЗОПРОВОДІВ
Царев В.Д. **74**

СЕКЦІЯ 17.
ФАРМАЦЕВТИЧНІ НАУКИ

МАРКЕТИНГОВИЙ АНАЛІЗ РИНКУ ПРОТИВІРУСНИХ ПРЕПАРАТІВ ЩО ЗАСТОСОВУЮТЬСЯ ПРИ ГРВІ
Приходько В.В. **76**

СЕКЦІЯ 18.
ФІЗИКО-МАТЕМАТИЧНІ НАУКИ

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ТЕПЛОПРОВОДНОСТИ
Жанахметова М.М. **79**

-4-

ОНТОЛОГІЧНЕ ПРЕДСТАВЛЕННЯ РЕЄСТРУ МЕТОДИЧНИХ РЕСУРСІВ КАФЕДРИ

Сьогодні розвиток інформаційних технологій приводить до зростання обсягів інформації. Відповідно виникають завдання, пов’язані з інтеграцією корпоративної інформації, отриманою з різних джерел та у різних форматах, з метою зручного та швидкого пошуку та подання користувачеві.

До інформаційних ресурсів кафедри можна віднести всю належну інформацію включаючи окремі документи і масиви документів, які належать до наукового, навчально-методичного та організаційного напрямків роботи кафедри.

До методичних ресурсів кафедри відноситься величезний пласт документів різних напрямків та різних форматів, пов’язаних з організацією навчального процесу [1]:

- навчальних планів; робочих навчальних планів;
- програм навчальних дисциплін, робочих програм кредитних модулів, плани практичних і семінарських занять, критерії оцінювання рейтингу студентів;
- підручники, навчальні посібники, конспекти лекцій;
- навчальні та методичні посібники до лабораторних робіт, курсових проектів,
- варіанти індивідуальних семестрових завдань, теми курсових проектів/робіт
- засоби діагностики для поточного та семестрового контролю результатів навчання з КМ та критеріїв оцінювання;
- завдання для проведення комплексних контрольних робіт з навчальних дисциплін та критеріїв оцінювання рівня підготовки студентів для проведення акредитації освітньої програми, моніторингу залишкових знань і вмінь;
- навчально-методичні матеріали дистанційного навчання (автоматизовані навчальні комплекси: відео-лекції, електронні підручники та практикуми, віртуальні лабораторні роботи, засоби тестового поточного контролю; методичні рекомендації щодо особливостей організації дистанційного і змішаного навчання тощо).

Сюди також можна віднести:

- комплект матеріалів щодо ліцензування, акредитації спеціальності;
- матеріали для абітурієнтів, програми вступних іспитів в магістратуру, аналіз результатів вступної компанії, тощо.

Реєстр інформаційних ресурсів – це інформаційна система, яка поводить реєстрацію, обліку, накопичення і зберігання відомостей про інформаційні ресурси та надання доступу до цих ресурсів користувачам системи [2]. По суті реєстр є каталогом у якому зберігається опис, розміщення, умови доступу, інформація про власника, формат, правами на коригування ресурсів та ін.

Сьогодні у якості сучасної моделі, яка використовується для структурування інформаційних потоків використовують онтологію. Онтологія дозволяє побудувати концептуальну модель предметної області, встановити різні типи зв'язків між класами та даними предметної області.

Для кращого розуміння вище сказаного розглянемо онтологію реєстру методичних ресурсів кафедри, опис класів та їх метадані(рис.1).

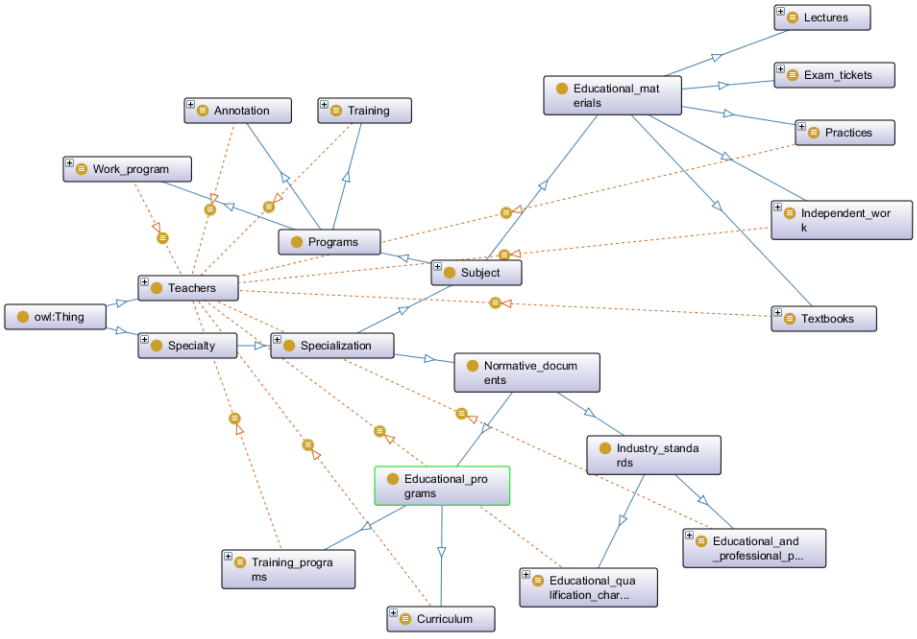


Рис. 1. Онтологія реєстру методичних ресурсів кафедри.

-6-

Для опису предметної області та структуризації методичних ресурсів використано наступні класи: спеціальність, викладачі, предмет, нормативні документи, галузеві стандарти, навчальні матеріали та програми, кожен з яких містить дані з різним смисловим значенням. Клас «Спеціальність» містить перелік спеціальностей, що є на кафедрі. Цей клас використано для зручності використання пошукової системи. Клас «Викладачі» містить список викладачів кафедри, кожен з яких є автором певного ресурсу, наприклад, переліку питань до іспиту. Клас «Предмет» містить перелік дисциплін базової підготовки. Клас «Нормативні документи» складається з двох підкласів «Навчальні плани» та «Навчальні програми». У класі містяться ресурси, які відносяться до опису роботи навчального процесу на кафедрі. Клас «Галузеві стандарти» ділиться на два підкласи «ОКХ» та «ОПП». Для зберігання матеріалів необхідних для студента створено клас «Навчальні матеріали», який у свою чергу складається з підкласів: «Екзаменаційні білети», «Самостійні роботи», «Лекції», «Практичні завдання», «Методички». Клас «Програми» містить три підкласи: «Анотації», «Навчальні програми», «Робочі програми», які містять характеристику про дисципліну та технологію її викладання.

Очевидно, що джерелом відомостей про інформаційний ресурс, є його метадані. Саме вони і повинні зберігатися у реєстрі. Основними метаданими даної онтології є: тип файлу, мова документу, дата публікації, автор, курс, відношення до предмету та належність до спеціальності. Метадані є основними критеріями для пошуку по онтології. За допомогою цих характеристик відбувається прив’язка одного класу до іншого.

Саме онтологія може бути певним джерелом знань та дозволяє під’єднати інформаційний пакет навчальної дисципліни до плану та визначає насиченість курсу необхідними матеріалами. Запити до онтології дозволять виконати пошук та проаналізувати насиченість дисциплін відповідним методичним забезпеченням.

Перевагою онтології як способу представлення знань є їх формальна структура, яка спрощує комп'ютерну обробку інформації.

Онтологія наглядно та зручно організовує групування інформаційних ресурсів. Крім того, зазвичай, онтологія зберігається у owl-форматі, який зручно використовувати при подальшій роботі з додатками.

Список використаних джерел:

1. Тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського: 5.4. Навчально-методичне забезпечення навчальних дисциплін

2. Ковтанюк Ю.С., Макарченко П.М., Дубровіна Л.А. Описування та організація зберігання електронних інформаційних ресурсів органів державної влади, органів місцевого самоврядування, підприємств, установ і організацій: методичні рекомендації / Укрдержархів України, УНДІАСД : Київ, 2010. – 30 с.

3. Котило М.О., Сєров Ю.О. Моделювання консолідованого інформаційного ресурсу для комунікативної взаємодії користувачів веб-сторінки кафедри / Реєстрація, зберігання і обробка даних. 2013. Т. 15, № 4. С. 23-31.